Multilingual Sentiment Analysis on Social Media

Erik Tromp

July 2011

Master's Thesis

Submitted in partial fulfillment of the
requirements for the degree of Master of Science

Multilingual Sentiment Analysis on Social Media

Erik Tromp (0608320)
e.tromp@student.tue.nl

Department of Mathematics and Computer Science
Eindhoven University of Technology

July 16, 2011

Abstract

The world wide web and more specifically social media are showing tremendous growth over recent years. The connectivity social media provide the world with allows users to more easily share experiences and influence each other through providing sentiment. The large volume of this sentiment calls for automated ways of interpretation to quickly gain insights and react as quickly as possible.

We investigate automated sentiment analysis on multilingual data from social media. As social media more and more connect the entire world, there is an increasing importance to analyze multilingual data rather than unilingual data. The automated sentiment analysis we perform extracts opinions from the relatively short messages placed on social media in multiple languages.

We present a four-step approach to perform sentiment analysis. Our approach comprises language identification, part-of-speech tagging, subjectivity detection and polarity detection. For language identification we propose an algorithm we call LIGA which captures grammar of languages in addition to occurrences of characteristics. For part-of-speech tagging we use an existing solution called the TreeTagger, developed at the University of Stuttgart. We apply AdaBoost using decision stumps to solve subjectivity detection. For polarity detection we propose an algorithm we call RBEM which uses heuristic rules to create an emissive model on patterns.

We extensively experiment with our four-step approach on the level of a single step as well as on the level of the complete process. Our single step comparisons show that our solutions perform better than other competitor solutions and baselines. We compare against approaches typically shown to be state-of-the-art in related work and baselines that are either general machine learning baselines such as a majority class guess as well as domain-specific baselines such as a prior polarity classifier. The experiments at the level of the complete process show that each step adds to accurately solving the sentiment analysis problem. We show that by leaving a step out of the process we obtain less promising results and report the overall accuracy of our sentiment analysis being 69.2%. We explicitly quantify the propagation of errors made by each of our steps to show where improvement can be easily made and what consequences this might have. We show that having an error rate up to 20% in our presented sentiment analysis approach still yields results better than more generic baselines.

Sentiment analysis can be used as an automated means to perform marketing research. The kind of marketing research currently addressing sentiment analysis uses traditional surveys to explicitly ask respondents for their opinion. We try to map the results of our automated sentiment analysis on the results of a traditional survey. We identify inherent problems that arise when making such a mapping. We investigate making an alignment using natural language as data source for both social media and the traditional survey and additionally experiment with mappings when our traditional survey uses score-based ratings to express sentiment.

We highlight possible applications of automated sentiment analysis and direct at possible future work. We show examples of views that can be created to gain insights into sentiment and state what can be improved or extended in the future. We implement an extendable framework realizing the theory presented in this work. We describe how this framework can be used, modified and extended to potentially incorporate the improvements we suggest for future work. We also demonstrate possible application areas in which our framework can provide solutions in the sense of insightful views.

Acknowledgements

First and foremost I would like to thank Mykola Pechenizkiy, my supervisor for this project. His devoted effort in guiding me throughout not only this thesis but also the preliminary work has greatly benefit the process. Thanks to Multiscope for allowing my creative mind the freedom it got. Particular thanks to Renzo de Hoogen from Multiscope for guiding me within the company and to Karen Visser from Multiscope for cooperatively creating and deploying the survey used in this work.

My signs of gratitude go out to my family – my father Ton, mother Ingrid and brother Paul – who have been supporting me since the moment I was born. More recently I have been receiving tremendous support from my fiancée Margit and my mother and sister in-law, Anneke and Dieuwertje, who lovingly provided me with a second home.

Without these people, this thesis would not exist.

Eindhoven
July 16, 2011

Erik Tromp

Contents

List of Tables

List of Figures

1 Introduction

In this thesis we study the problem of sentiment analysis. Sentiment analysis is the process of automatically determining sentiment expressed in natural language. The term *sentiment* is a broad one, varying from emotions to personal experiences. In this thesis we are concerned with a particular subset of sentiment, the opinions. The problem we hence study is to determine the general opinion expressed in texts written in natural language.

In Section 1.1 we motivate why studying sentiment analysis is interesting. In Section 1.2 we more clearly state what sentiment analysis actually is by presenting a problem formulation which we subsequently use and answer throughout this thesis. The main results presented in this thesis are summarized in Section 1.3. The further outline of this thesis is presented in Section 1.4.

1.1 Motivation

Sentiment analysis is an increasingly active problem. Consumers use the web as an advisory body influencing their view on matters. Knowing what is said on the web allows to react upon negative sentiment and to monitor positive sentiment. Doing this by hand however is a tedious if not impossible task. With the growth of the web and especially social media, more and more attention is paid to the automated retrieval of sentiment from all kinds of sources.

Social media stimulate the information overload present on the web. Twitter[1] produces 50 million messages per day in 2010[2], an average of 600 tweets per second. The ease with which such a tweet can be placed stimulate the spread of people's opinions. This spectacular volume calls for an automated interpretation to flexibly and quickly respond to shifts in sentiment or rising trends.

As social media cover almost the entire world, the sentiment expressed by users of social media is written in a multitude of languages. The social media connect the entire world and thus people can much more easily influence each other. The presence of for example free online translation services makes the difference in language less important in this setting, which is a motivation for the need to cover multiple languages. Moreover, as more and more people share their opinion on social media, insights into sentiment can be obtained much quicker when performed automatically. This way one can almost directly respond on phenomena arising in society than traditionally.

A recent real-life example motivating the need for sentiment analysis is that of a Dutch comedian named Youp van 't Hek and T-Mobile. Comedian van 't Hek's son had a problem with his iPhone not working well with T-Mobile. The customer service did not help much via telephone after which van 't Hek decided to post a Twitter message about his problem. Little attention was paid to this tweet which triggered more complaints by van 't Hek and more importantly, his followers as they were influenced by van 't Hek's sentiment, unleashing a burst of negative sentiment on T-Mobile. Research performed by Buzzcapture showed that the image damage to T-Mobile resulting from this burst of negative sentiment lies somewhere between 200.000 and 300.000 euros[3]. Through sentiment analysis, the burst of negative sentiment started by van 't Hek could have been anticipated as shifts in sentiment could have been discovered through monitoring sentiment. Moreover, if this sentiment analysis is performed automatically, the anticipation can occur much more adequately, perhaps even avoiding bursts of negative sentiment in the first place by quickly responding when the amount of negative sentiment is still manageable.

A practical motivation for performing sentiment analysis is that of Multiscope[4] with which this project is cooperatively executed. This company carries out traditional online surveying and is looking for ways to complement this process with automated processes. The resources used in this project are partially made

[1] http://www.twitter.com/ – A microblogging platform

[2] Numbers based on http://blog.twitter.com/2010/02/measuring-tweets.html

[3] Numbers taken from http://buzzcapture-com.pressdoc.com/9051-youp-van-t-hek-brengt-t-mobile-reputatieschade-toe-via-twitter – measured by analyzing the differences in sentiment and translating this to financial numbers

[4] http://www.multiscope.nl – A company performing online marketing research

available by Multiscope. More specifically, the survey used in our alignment with traditional surveying is constructed, distributed and made available to this project by Multiscope.

1.2 Problem Formulation

Sentiment analysis can be performed at different levels of granularity with different levels of detail. We perform sentiment analysis on social media in which a single message typically consists of one or two sentences. Supported by this observation, the type of granularity we study is the sentence level. Other granularity levels can be the document level [Sindhwani and Melville, 2008, Pang and Lee, 2004], word level [Hatzivassiloglou and McKeown, 1997] or the phrase level [Riloff et al., 2003]. The level of detail typically goes into determining the *polarity* of a message, which is what we investigate as well. A more detailed approach could be to determine the emotion expressed [Potena and Diamantini, 2010] in addition to the polarity.

The problem we investigate thus takes a message m as input and produces a polarity indication p as output. The message m originates from one of the following social media: *Twitter, Facebook*[5]*, Hyves*[6]. The language of m is not known beforehand and we are interested in messages written in a closed set of multiple languages $l_1, .., l_n$ rather than only regarding one language, for example English.

Conceptually, we can regard the problem of sentiment analysis as shown in Figure 1 where we have a stream of unstructured texts originating from social media as input. Though we use different social media, we regard all of our input as plain text. Determining the opinion of a message can be as extensive as regarding *objective* (not expressing any sentiment), *positive* (expressing positive sentiment), *negative* (expressing negative sentiment), *neutral* (expressing sentiment but neither positive nor negative) and *bipolar* (expressing both positive and negative sentiment) messages. We however assume that messages that are neutral are also objective. Moreover, as the messages placed on social media are very short, we also assume that only one polarity is prevailingly expressed in a message, we thus do not regard the bipolar class as we assume that a message containing two polarities always favors one of the two. Additionally, we associate the language a text is written in with each text, allowing for more fine-grained sentiment analysis by exploiting knowledge on the language a message is written in. Knowing the language of each message also allows for segmenting based on language. Though not our goal in this project, the results of such a sentiment analysis can be aggregated in different ways to obtain high-level statistics. Combining profiling information such as age or gender allows for a more fine-grained sentiment analysis on a segment-level.

Concisely stated, the answer to the problem we solve is a hard label being one of *objective*, *positive* or *negative*. The formulation of the problem itself is as follows: *Given a message written in an unknown language to be determined, does this message contain sentiment and if so, what is the polarity of this sentiment?*

1.3 Our Approach & Main Results

We introduce a four-step approach for automated sentiment analysis. For language identification we propose a new algorithm called LIGA. This algorithm uses a graph formalism to incorporate grammar into its model. For part of speech tagging (POS-tagging) we use existing work having publicly available models for different languages. For subjectivity detection we propose using AdaBoost ensemble learning with decision stumps as weak learners. Finally, we introduce a new algorithm called RBEM for polarity detection. This algorithm uses eight heuristic rules defined on patterns to determine the polarity of a text.

We extensively experiment with our four-step approach and each of its separate steps. We show that our approach to each of the four steps outperforms other competitor approaches and baselines. We also

[5] http://www.facebook.com/ – The world's largest social network
[6] http://www.hyves.nl/ – Holland's largest social network

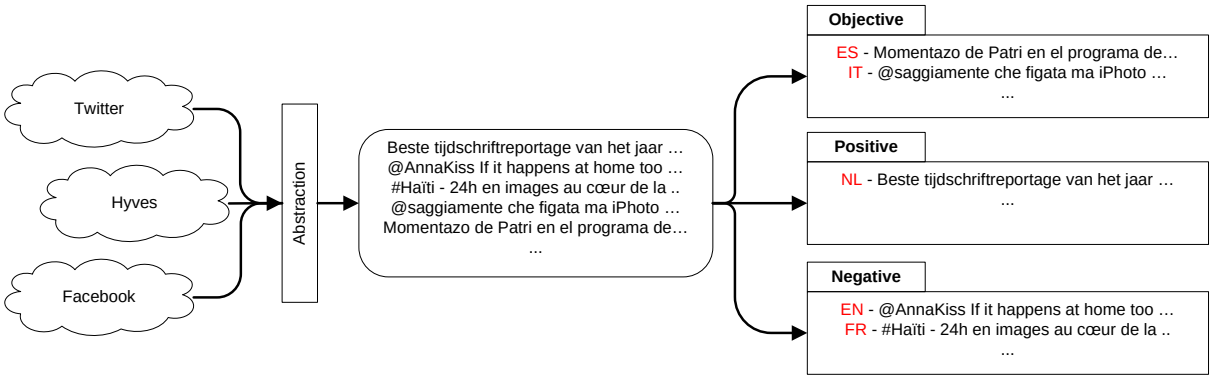


Figure 1: The conceptual idea of sentiment analysis. The input is a stream of social media messages in different languages. The output is a crisp classification into objective, positive or negative along with the language of a message which is an additional requirement also allowing for more specialized approaches.

show that each step aids to better solving the sentiment analysis problem by leaving each step out and comparing it against not leaving it out. For practical reasons one may want to use faster or simpler approaches at the cost of introducing higher error rates. We quantify the propagation of these errors along the four-step pipeline. This shows the effect of using approaches with lower accuracy with respect to the approaches we propose.

We attempt to create an alignment between the results of our four-step sentiment analysis results' against those of a traditional survey. We show that the results of automated sentiment analysis are difficult to map onto the traditional survey's results. We identify problems inherently present with the mapping of the automated sentiment analysis on the traditional survey and propose alternatives to overcome these issues. We investigate making an alignment using natural language from social media against also using natural language for our traditional survey. We additionally investigate what it implies if we make a mapping when using a score-based rating to express sentiment for our traditional survey rather than using natural language

We created a framework implementing the theory presented in this thesis. Though not publicly available, this framework is outlined in Appendix A and allows to automatically extract data from social media, thus providing input for the actual sentiment analysis process. It also allows for easy addition of insightful views that differ for each purpose. In our specific case, the sentiment analysis process consists exactly of the four steps we present in this thesis but any other process can be "plugged in" or components of our sentiment analysis process can be altered, removed or supplemented with other components.

Sentiment analysis is a problem that can be applied in many areas. We highlight five possible application areas where our four-step approach can be used to support daily business, specifically in the case of using it to support in answering marketing questions in Appendix C. We demonstrate five settings as example applications of sentiment analysis. We discuss the strengths and weaknesses of using sentiment analysis over traditional surveys in each use case.

1.4 Outline

In Section 2 we introduce our four-step approach to the sentiment analysis problem and refer to related work. This four-step approach entails *language identification (LI)*, *part-of-speech (POS) tagging*, *subjectivity detection* and *polarity detection*. For language identification we present an algorithm called LIGA (Language Identification through a Graph based Approach) in Section 2.1.3. For POS-tagging we use Stuttgart's TreeTagger which we briefly describe in Section 2.2. Subjectivity detection we approach using AdaBoost, which we describe in Section 2.3. We conclude our approach by describing polarity detection in Section 2.4. For polarity detection we introduce an algorithm called RBEM (Rule-Based Emission Model) in Section 2.4.2.

We extensively experiment with our four-step approach in Section 3. We perform comparative exper-

iments where each individual step is compared against other approaches in Section 3.3.1 to find the empirically best performing local solution. We also compare the complete four-step approach against more generic approaches where one of the four steps is left out in Section 3.3.2. The propagation of errors made at each step shows the importance to have each step as accurate as possible. We analyze the propagation of errors in Section 3.3.3.

In addition to these comparative experiments, we investigate making an alignment between our sentiment analysis' results and those of a traditional survey in Section 3.4. We identify problems that inherently occur with this mapping in Section 3.4.1. We assume a mapping between the two exists and investigate the alignment when using natural language as data for both our sentiment analysis as well as our traditional survey in Section 3.4.2. With this alignment we evaluate the sentiment distributions and quantify differences in sentiment distributions as more errors are made. We also investigate differences in the data sources. We additionally investigate mappings when our traditional survey's data consists of score-based ratings rather than natural language in Section 3.4.3. We do this on the segment level and global level and analyze what the influence is of training our mapping on our survey data. The figures we use to analyze the alignment are presented in Appendix B. We summarize our traditional survey experiments in Section 3.4.4.

We conclude by presenting the main results and achievements in Section 4. We highlight the main contributions of our work in Section 4.1. We also direct at possible practical applications in Section 4.2 in which we also hint at possible applications of our work and briefly summarize demonstrative application examples presented in Appendix C. Finally we point out directions for potential future work in Section 4.3.

We present five examples demonstrating application areas of sentiment analysis in Appendix C. We consider applying sentiment analysis to monitor sentiment in Appendix C.1. We give an example how sentiment analysis can be used to create benchmarks in Section C.2. Public polling shows the sentimental attitude of a nation or society towards a specific entity. We present an example how sentiment analysis can be used to create election polls in Section C.3. We extend our election polling example by incorporating forecasting techniques to predict the actual election's outcome in Section C.4. Web care is an increasingly important aspect of customer relationship management and we present a use case on applying sentiment analysis to support in providing adequate customer web care in Section C.5.

Technical specifications of the software implementing the framework on which we base our analysis can be found in Appendix A. This appendix comprises implementation specifications, database setup and configuration instructions. In addition to software, a traditional survey is created for use in our experiments. This survey is presented in its entirety in Appendix D.

2 Four-Step Approach for Sentiment Analysis

The approach we take to solve the sentiment analysis problem consists of four steps; *language identification*, *part-of-speech tagging*, *subjectivity detection* and *polarity detection*, as shown in Figure 2. Information is retrieved from social media. As this information differs in format we use an abstraction layer such that all input looks the same to our sentiment analysis process. This sentiment analysis process in turn takes as input an unstructured, short text of which no prior knowledge is known, not even the social medium from which it originates. At each step more information is added and as output we either get a polarity or that the text is objective. This implies that only subjective texts are processed by the polarity detection step as objective texts do not have any polarity. The rationale behind using four steps is that we can more specifically construct models in a later step when knowledge from earlier steps is present. The separation of subjectivity and polarity detection is inspired by [Pang and Lee, 2004]. The use of each step is experimentally investigated in Section 3. Each step tackles a subproblem on its own and we describe each problem separately and present our solution to it.

The approaches we next present are implemented in an extendable framework discussed in more detail in Appendix A. This framework allows to crawl and scrape social media for data that can serve as input for the sentiment analysis process. The sentiment analysis itself in our setting consists of the four steps we next describe but due to the nature of the framework, more steps can be added without jeopardizing the other steps. For example, an emotion classifier – a more elaborate case of sentiment analysis where the emotion of a message is automatically extracted – can be added as an additional step. Alternatively, steps can be removed or supplemented and even a completely different (sentiment analysis) process can be used instead of what we use without having to reconstruct the data collection.

Each of the separate steps of our framework can also be extended, modified or replaced to suit different purposes. As the input of one step is the output of another step, there is no dependency between the actual operational properties of the steps but only on the input and the output. This means that any step can be replaced or extended in any way as long as the input and output remain the same. As an example, it is possible to replace our approach for subjectivity detection with a Naive Bayes algorithm also performing a two-way classification into subjective or objective.

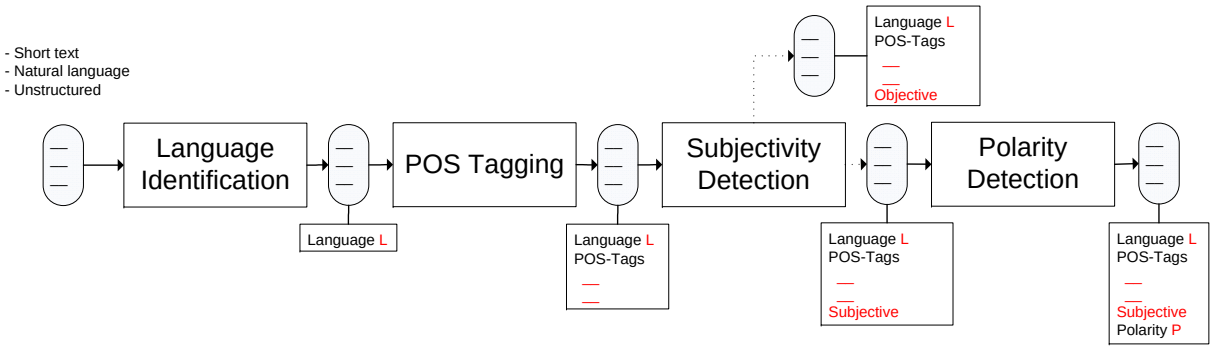


Figure 2: The four-step approach for sentiment analysis. The input consists of unstructured texts. First language identification determines the language after which POS-tagging enriches the text with additional features (POS-tags). The subjectivity detection determines if the text contains subjective aspects. If not, the text is objective, otherwise the text's polarity is determined by the polarity detection.

2.1 Language Identification

Language identification is the first step in our process. Knowing the language gives us the opportunity to apply more specific models in succeeding steps. Additionally, we can filter out those languages which we are not interested in. We first formulate the problem of language identification in Section 2.1.1 after which we describe related work in Section 2.1.2 and present our approach in Section 2.1.3.

2.1.1 Problem Formulation

Language identification is a supervised learning task. Given some historical or training data in which for each text t we know a label l, the language in which this text is written, our goal is to learn a model such that given some previously unseen text we can say as accurately as possible in which language this text is written.

In our study, we do not consider cases when for a text written partly in one language and partly in some other language someone would like to get both labels as an output. We also do not consider language groups or any other dependencies between the language labels, thus yielding a more specific case of the generic language identification problem.

2.1.2 Related Work

The N-gram-based approach for language identification [Cavnar and Trenkle, 1994] chops texts up in equally-sized character strings, N-grams, of length n. It is assumed that every language uses certain N-grams more frequently than other languages, thus providing a clue on the language the text is in. Experimental studies in [Cavnar and Trenkle, 1994] suggest that using trigrams (at the character level) generally outperform using other sized N-grams.

The idea behind the N-gram-based approach is borrowed from [Dunning, 1994] where using Markov models for language identification was considered. This approach however lacks the intuition the N-gram approach has and requires more time for training a model and for classifying a new text.

A simple, straightforward approach is to use word frequencies. One variant is to use short words [Prager, 1999] as they occur regularly in a language and usually differ per language. Another variant is to use the most frequently occurring words [Martino and Paulsen, 1996, Cowie et al., 1999] for the same rationale.

A compression-based approach for language identification was proposed in [Harper and Teahan, 2001]. Labeled data is compressed using a so-called prediction by partial matching (PPM) approach to construct language models. An unlabeled text is also compressed and the number of bits required to encode this new document is compared to the of number bits used in the language models. The likeliness of a text belonging to a language model is computed using entropy as a similarity measurement.

2.1.3 Our Approach

We give a compact description of our approach for language identification. For a more elaborate description we refer to [Tromp and Pechenizkiy, 2011]. This approach entails an algorithm using a graph formalism and N-grams that specifically targets identifying the language of messages present on social media. Given the problem formulation presented in Section 1.2, restated below with a bold font on the part answered by language identification concerning the determination of the language.

Given a message written in **an unknown language to be determined**, *does this message contain sentiment and if so, what is the polarity of this sentiment?*

In our approach we want to utilize not only word presence and occurrences but also their ordering. To capture the ordering of words, we create a graph model on labeled data. The labels of its vertices represent the presence of words in a given language. The weights of the vertices represent the frequencies of words in a given language. The crucial part is in the presence and weights of the edges, which try to capture the grammar of a language.

In this particular case we only capture the ordering as a grammatical aspect. We use character N-grams in our approach. We will thus not truly capture word information but N-gram information. Next, we

give the preliminaries, the methodology to learn LIGA and to classify unlabeled texts. We additionally provide a methodology to filter out messages written in languages not present in our model.

PRELIMINARIES

We extend a basic graph $G = (V, E)$ with a labeling function $\mathcal{L} : V \to L$. This labeling function assigns to each vertex $v \in V$ a label $l \in L$ uniquely identifying the vertex. Let *Lang* denote all languages present in our training set, then the function $\mathcal{W}_v : V \times Lang \to \mathbb{N}$ assigns for each vertex $v \in V$ and every language $l \in Lang$ a weight. For edges we have a similar function $\mathcal{W}_e : E \times Lang \to \mathbb{N}$. Since we will incrementally construct our graph, we may encounter that, for a label l of a vertex u to be added, $\exists_{v \in V} : v \neq u \wedge \mathcal{L}(v) = l$. We then say that $\mathcal{L}(v)$ is *defined*. We say $\mathcal{L}(v)$ is *undefined* in all other cases. We use the same notion of *defined* for $\mathcal{W}_v$ and $\mathcal{W}_e$.

Using the mentioned extensions, we represent a graph as the following quintuple.

$$G = (V, E, \mathcal{L}, \mathcal{W}_v, \mathcal{W}_e)$$

A labeled text t of which the language l is known is denoted as the binary tuple (t, l). An unlabeled text s will be denoted as the binary tuple (s, λ). We denote all N-grams of a text t as the ordered list $N_{n_t} = [g_1, g_2, ..., g_k]$ where n denotes the length of the N-grams. The order in N_{n_t} respects the order in which the N-grams occur in (t, l).

LEARNING A MODEL

Our goal is given a training set $\mathcal{T}$ consisting of labeled texts for every language in *Lang* to learn a model consisting of a single graph G.

For each text $(t, l) \in \mathcal{T}$ we construct the list N_{n_t}. For every $m \in N_{n_t}$ we create a vertex v with $\mathcal{L}(v) = m$, but only if $\mathcal{L}(v)$ is *undefined*. For every $e \in \{(u, v) \in V \times V : (\mathcal{L}(u) = m_i \wedge \mathcal{L}(v) = m_{i+1}) \Rightarrow (m_i, m_{i+1} \in N_{n_t})\}$ we create an edge e, but only if $e \notin E$. The weights are updated as follows:

$$\mathcal{W}_v(v, l) = \begin{cases} \mathcal{W}_v(v, l) + 1 & \text{if } \mathcal{L}(v) \text{ is } \textit{defined} \\ 1 & \text{otherwise} \end{cases}$$

$$\mathcal{W}_e(e, l) = \begin{cases} \mathcal{W}_e(e, l) + 1 & \text{if } \mathcal{W}_e(e, l) \text{ is } \textit{defined} \\ 1 & \text{otherwise} \end{cases}$$

When we add a node or edge (i.e. when $\mathcal{L}(v)$ or $\mathcal{W}_e(e, l)$ is *undefined* respectively), we initialize the weights of all languages for that vertex or node to 0 before applying the weight updates. When applying the aforementioned definitions for all $(t, l) \in \mathcal{T}$ we get our graph $G = (V, E, \mathcal{L}, \mathcal{W}_v, \mathcal{W}_e)$. We illustrate this by an example. Consider the following two texts of which the first is in Dutch (NL) and the second is in English (EN).

Example 1
$(t_1, NL) =$ *is dit een test*
$(t_2, EN) =$ *is this a test*

We then first create the ordered lists of N-grams. When using trigrams ($n = 3$) we get the following, where a space is denoted by a dot ·.

$N_{3_{t_1}} =$ [IS·, S·D, ·DI, DIT, IT·, T·E, ·EE, EEN, EN·, N·T, ·TE, TES, EST]
$N_{3_{t_2}} =$ [IS·, S·T, ·TH, THI, HIS, IS·, S·A, ·A·, A·T, ·TE, TES, EST]

We next start constructing the graph. For each $n \in N_{3_{t_1}} \cap N_{3_{t_2}}$ we add a vertex v to our graph, having $\mathcal{L}(v) = n$. For example, for the first element in $N_{3_{t_1}}$ we will create the vertex v having $\mathcal{L}(v) = is\cdot$. For the first element in $N_{3_{t_2}}$ we will not add a new vertex as $\mathcal{L}(v)$ is defined. In our example, for vertex v (having $\mathcal{L}(v) = is\cdot$) we will have $\mathcal{W}_v(v, NL) = 1$ and $\mathcal{W}_v(v, EN) = 1$ as $is\cdot$ occurs once in both the Dutch as well as the English text.

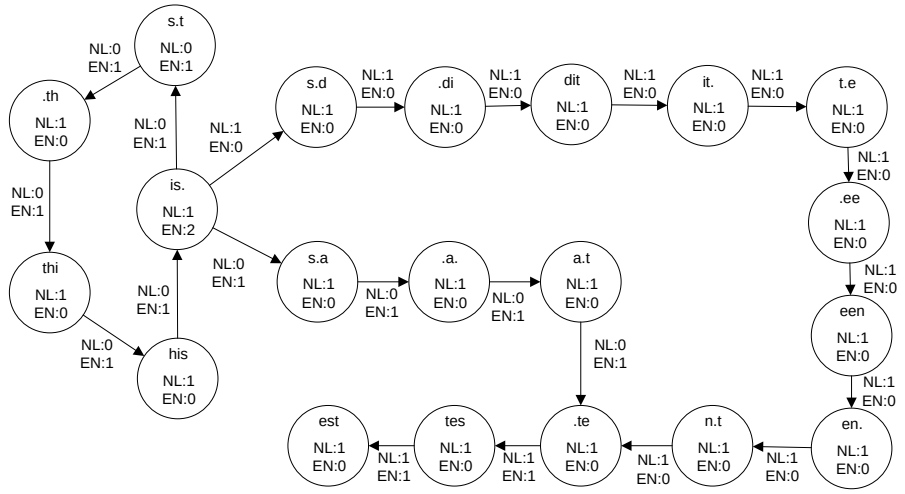


Figure 3: The graph resulting from Example 1. Each N-gram is a vertex of which the weights for each language are given inside it. The ordering of N-grams is captured by the weighted edges between them.

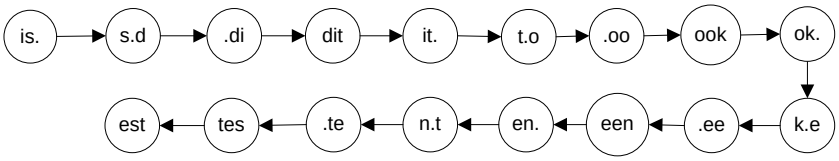


Figure 4: The path resulting from the unlabeled example.

We next add edges. We will have edges from for example v to u ($e = (v, u)$) where $\mathcal{L}(v) = is\cdot$ and $\mathcal{L}(u) = s \cdot d$, capturing the order between the first and second elements of $N_{3_{t_1}}$. Since this connection occurs only once, and only for the Dutch text, we have that $\mathcal{W}_e(e, NL) = 1$ and $\mathcal{W}_e(e, EN) = 0$.

Figure 3 shows the graph resulting from this example. The labels of the vertices are shown at the topmost position inside the vertices. The weights per language are listed with the vertices and edges.

Classifying a Text

Once we have constructed G, we can use it to classify unlabeled texts. To do so, we first need to transform an unlabeled text into something similar to G such that we can compare the two.

While for constructing G we use weights to indicate multiple occurrences of a given N-gram, for the unlabeled text we create multiple vertices – one for every occurrence. We thus in fact get a simple graph, a path $\pi = (V, E, \mathcal{L}, v_{start})$. Here $|V| = |N_{n_t}|$ and if $N_{n_t} = [n_1, n_2, .., n_k]$ then $E = \{(u, v) | \mathcal{L}(u) = n_i \wedge \mathcal{L}(v) = n_{i+1}\}$. The node $v_{start} \in V$ is the starting node of our path. To illustrate this, we consider the following (Dutch) text *is dit ook een test* of which we would not know the label in advance. The path π for this example is shown in Figure 4.

A crucial step is to compare π and G. We compute the so-called *path-matching scores* for each language $l \in Lang$. Conceptually, this is done by 'laying' the path over the graph and measuring its similarity for each language. Since all languages we have knowledge of are present in a single model, we can compute the scores for all languages in one go.

To keep track of the matching scores of our languages, we maintain a scoring function $\mathcal{PM} : Lang \to \mathbb{R}$ assigning a rational number to every language $l \in Lang$. Initially, we say that $\mathcal{PM}(l) = 0$ for all $l \in Lang$. Let $G = (V, E, \mathcal{L}, \mathcal{W}_v, \mathcal{W}_e)$ be the graph representing our model and $\pi = (V', E', \mathcal{L}', v_{start})$ be our labeled path. In order to keep our method robust with respect to differing quantities of labeled data for different languages, we will normalize the scores. Let $\Sigma_v = \Sigma_{v \in V}(\Sigma_{l \in Lang}(\mathcal{W}_v(v, l)))$ be the total sum of all weights contained in all nodes in G. Also, let $\Sigma_e = \Sigma_{e \in E}(\Sigma_{l \in Lang}(\mathcal{W}_e(e, l)))$ be the sum of all weights contained in all edges in G. We then start traversing our path π in G starting with node v_{start}. Let v'_i denote the node we are currently visiting (initially, $v'_i = v_{start}$). We try to find a $v \in V$ such that $\mathcal{L}(v) = \mathcal{L}'(v'_i)$. Note

$$\forall_{l \in Lang} : \mathcal{PM}(l) = \begin{cases} \mathcal{PM}(l) + \frac{\mathcal{W}_v(v,l)}{\Sigma_v} & \text{if } \exists_{v \in V} : \mathcal{L}(v) = \mathcal{L}'(v'_i) \\ \mathcal{PM}(l) & \text{otherwise} \end{cases} \quad (1)$$

$$\forall_{l \in Lang} : \mathcal{PM}(l) = \begin{cases} \mathcal{PM}(l) + \frac{\mathcal{W}_e(e,l)}{\Sigma_e} & \text{if } \exists_{e \in E} : (\exists v, w \in V : \mathcal{L}(v) = \mathcal{L}'(v'_i) \wedge \\ & \quad \mathcal{L}(w) = \mathcal{L}'(v'_{i+1}) \wedge e = (v, w)) \\ \mathcal{PM}(l) & \text{otherwise} \end{cases} \quad (2)$$

that there is at most one such node but there may be none. We update the matching scores according to (1). In this step we account for the occurrence of a given N-gram from π in G.

The next step is to account for the order, which is only performed if we were able to find a $v \in V$ such that $\mathcal{L}(v) = \mathcal{L}'(v'_i)$. We find the only edge $e' \in E'$ that has our previous node v'_i as source, if any (since the last node has no outgoing edges). We thus find the edge $e' = (v'_i, v'_{i+1})$. We then update the matching scores according to (2).

We have now accounted for the order between two N-grams present in our path whose label is given in nodes v'_i and v'_{i+1}. We next continue traversing our path by performing (1) again for v'_{i+1} after which we can apply (2). This process continues until we find a node $v'_{end} \in V'$ such that $\neg(\exists_{v' \in V'} : (v'_{end}, v') \in E')$; this is the ending node not having any outgoing edges.

When we have matched the entire path onto the graph, we need our function $\mathcal{PM}$ to determine the language. For each language, $\mathcal{PM}$ maps it onto a value in $\mathbb{R}$. More specifically, since the weights of the nodes accounted to $\mathcal{PM}$ are normalized and so are the weights of the edges, we end up with a score in $[0, 2]$ (when our model is isomorphic to our path for a given language, we get a score of 1 for the nodes and 1 for the edges). We now say that the text of which we were identifying the language is in language $l = argmax_{l \in Lang} : \mathcal{PM}(l)$.

Filtering Out Unknown Languages

As in our setting we are only interested in a subset of all languages used on social media, we want to filter out those messages that we are not interested in. For example, our succeeding steps may only have models for languages $Lang = l_1, .., l_n$, any language $l' \notin Lang$ can not be processed by succeeding steps and hence needs to be filtered out. To filter out such messages written in languages not present in our model, we add an additional threshold. Consider the case where we have a model containing information on languages $L = \{l_1, .., l_n\}$. If we now encounter a message m whose actual language is r, having $r \notin Lang$, we still assign scores for each language $l_i \in Lang$. As the algorithm then dictates, the language with the highest score is the resulting class but since $r \notin Lang$ this can never be the right class. We therefore need such a threshold.

The threshold is defined as dividing the resulting score by the message's length in characters. For our experiments, we use a threshold of 0.00125 which showed the most promising results during a small manual inspection of a simple experiment using different thresholds. Whenever no language obtains a score higher than this threshold, we label the message's language to be *UNKNOWN* and disregard it. Note that the threshold we use stems from a very small-scaled experiment and hence more extensive experiments will show more promising thresholds. For our purposes however, this threshold shows sufficient performance in the small-scaled experiment in which we compared different thresholds.

2.2 Part-of-Speech Tagging

Part of speech tagging is the second step in our approach. We use this step merely to expand our feature space. Related work [Pang et al., 2002, Pang and Lee, 2008] shows that using just word tokens for subjectivity or polarity detection – the succeeding steps in which the expanded feature space can be used – yields similar or better results than using POS-tags with for example Bayesian and SVM approaches. As such, the accuracy of this step is expected not to be as crucial as that of other steps since we can choose using tokens instead of POS-tags. Nevertheless, a higher accuracy is more beneficial. Both subjectivity detection as well as polarity detection make use of the POS-tags of a message.

We first describe the part-of-speech tagging problem more formally in Section 2.2.1. We then point to related work in Section 2.2.2 after which we describe the approach we take to solving part-of-speech tagging in Section 2.2.3.

2.2.1 Problem Formulation

Part-of-speech tagging is the process of assigning a grammatical group to a word. A model is trained on tagged data where for each word of a sentence, its POS-tag is known. Words are hence in context and one single word may have different POS-tags depending on the context. The task at hand is to assign POS-tags $[t_1, .., t_n]$ to each of the words $w_i \in W$ of a previously unseen sentence W containing just words.

POS-tags differ per language. We thus require to know the language in advance before applying POS-tagging. When we apply POS-tagging, we use different models for each language, thus supporting the different POS-tags used in each language.

2.2.2 Related Work

Different approaches to POS-tagging have been proposed. An elegant and simple approach is the Brill Tagger proposed in [Brill, 1992]. This tagger uses rules selected from a predefined set that define what POS-tag is assigned to what word. The tagger iteratively evaluates the attribution of each rule to the accuracy and uses only those rules that score best.

More prevailing in POS-tagging are stochastic approaches. Fully Bayesian approaches have been proposed [Goldwater and Griffiths, 2007] but hidden Markov models (HMMs) and its extension; conditional random fields (CRFs) are more commonly used [Cutting et al., 1992, Kupiec, 1992, Lafferty et al., 2001]. These stochastic approaches are typically supervised but there is also work on unsupervised and even semi-supervised approaches.

2.2.3 Our Approach

We solve POS-tagging by using Schmid's TreeTagger developed at the University of Stuttgart and introduced in [Schmid, 1994]. Markov model approaches to solving POS-tagging typically use potentially sparse data to estimate transition probabilities. Due to the sparsity of the data, these probability estimates may be relatively unreliable. The TreeTagger remedies this issue by not only estimating word probabilities but also incorporate a suffix model.

Instead of using Markov models, the TreeTagger uses decision trees where the leafs contain probability estimates of a tag occurring given x previous tags. In his work, [Schmid, 1994] uses trigrams and hence only considers the previous two tags, resembling a second order Markov model. To create the decision tree, a modified version of the ID3-algorithm [Quinland, 1983] is used where information gain is used as splitting criterion.

The strength of the TreeTagger however lies not within just exploiting decision trees where traditionally Markov models are used. In addition to just using (word) N-grams and their probability estimators, [Schmid, 1994] also uses a suffix tree. Such a suffix tree acts similar to the regular N-gram tree but only relevant suffixes up to a given length ([Schmid, 1994] uses a length of three) are used. If no match can be made against the N-gram tree, the suffix tree is used. Within this tree, first the last character of the respective word is looked up. Next its last-but-one character is looked up and finally its last-but-two character is looked up, in the case of a suffix tree of length three. It may occur that even a match against the suffix tree is not possible. In such a case, a default a priori POS-tag is used, defined by the prevailingly occurring tag for a word.

In our example presented in Section 2.1.3, we had the Dutch text *is dit ook een test*, the English equivalent is *is this also a test*. A possible output of the POS-tagger is shown in Example 2.

Example 2
The POS-tags for the message *is this also a test*.

Word	Tag	
is	VBZ	(Verb)
this	DT	(Determiner)
also	RB	(Adverb)
a	DT	(Determiner)
test	NN	(Noun)

The rationale behind using this tagger is that it contains publicly available models in numerous languages. The TreeTagger can output multiple tags with varying probabilities for a single word. We only use the most probable tag and disregard the others. The POS-tags will not answer any part of the problem formulation given in Section 1.2 but rather serve as additional information for succeeding steps.

2.3 Subjectivity Detection

Subjectivity detection is our third step. Once we know the language of a message and have determined its POS-tags, we identify whether a message expresses private state or not. If this is not the case, a message is said to be objective which is one of our labels and hence the message is not processed further. If a message does contain private state, it is said to be subjective and we pass such a message on to polarity detection to determine the polarity expressed.

The problem formulation is more concretely defined in Section 2.3.1. We refer to related work not only on subjectivity detection but also on polarity detection in Section 2.3.2 as these two processes are often combined into one. We present our approach on subjectivity detection in Section 2.3.3.

2.3.1 Problem Formulation

The goal of subjectivity detection is to determine whether a given message is subjective or objective. By subjective we mean a message containing any form of private state; a general term covering opinions, beliefs, thoughts, feelings, emotions, goals, evaluations and judgements [Quirk et al., 1985]. More formally, subjectivity detection is a supervised learning task taking a message m as input, providing a label $s \in \{\textit{Subjective}, \textit{Objective}\}$ as output. The message m can consist of just its words (tokens), just its POS-tags or both. Note that when a message is objective, it is not processed by the polarity detection step. If a message is subjective, the polarity detection step next identifies its polarity. Since we know the language a message is in through our earlier language identification step, we can apply language-specific models.

2.3.2 Related Work

Subjectivity detection is often combined with polarity detection, either as two separate steps or as one big step. We therefore refer to work on either subjectivity or polarity detection (or both).

One of the earliest works in this area is that of [Hatzivassiloglou and McKeown, 1997] who deduce polarity of adjectives using different conjunctive words. More elaborate work was done by [Pang et al., 2002] who compare numerous machine learning techniques for opinion mining. They continue their work in [Pang and Lee, 2004] and [Pang and Lee, 2008].

Other works are those of [Riloff et al., 2003, Wilson et al., 2005, Wiebe et al., 2005, Wiebe and Micalcea, 2006]. In these related works, the authors start from bootstrapping methods to label subjective patterns.

In their latest work, both subjectivity and polarity detection is performed and evaluated using these patterns along with high precision rules defined in their earlier works.

More recently attention is being paid to sentiment analysis on social media. Sentiment analysis on Twitter is researched by [Go et al., 2009, Pak and Paroubek, 2010] who use similar methodologies to construct corpora and analyze Twitter messages to determine their polarity. [O'Connor et al., 2010] use opinion mining on Twitter to poll the presidential election of the United States in 2008 and show how using Twitter opinion time series can be used to predict future sentiment.

2.3.3 Our Approach

The approach we take to solving subjectivity detection is inspired by the text categorization methodology *BoosTexter* [Schapire and Singer, 2000]. BoosTexter uses boosting to categorize texts. For our subjectivity detection we do not use BoosTexter but do use the boosting principle by means of the simpler *AdaBoost*, first introduced in [Freund and Schapire, 1995].

AdaBoost uses an ensemble of weak learners to create a strong learner. It does this on a per-round base where each round, those entries that were misclassified get more attention next round by increasing the weight of these entries. This process continues until either the maximum number of rounds is reached or the weighted error is more than 50%.

The weak learners we use are decision stumps, decision trees of length 1. These stumps can be seen as rules of the form *if f present then label = a else label = b* where f is a (binary) feature and a and b are possible labels. In our case we only regard presence of features and consider a biclass classification problem. The model-weighting function we use (see [Freund and Schapire, 1995]) is $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$ where ϵ_t is the classification error at iteration t.

Supported by the results of Section 3.3.1, we use 50 weak learners and POS-tags combined with a prior polarity lexicon for features. The prior polarity lexicons contain positive, negative and negation words. In addition to the presence of POS-tags we have one additional feature indicating the presence of exactly one positive word (weak positiveness) and one feature indicating the presence of multiple positive words (strong positiveness). We also have two identical features for the negative case. Whenever a positive or negative word is directly preceded by a word from the negation list, its polarity is flipped.

Taking our problem formulation posed in Section 1.2 into account, which is restated below, we solve the bold part of determining whether sentiment is contained with in the message or not.

Given a message written in an unknown language to be determined, **does this message contain sentiment** *and if so, what is the polarity of this sentiment?*

2.4 Polarity Detection

The last step of our process is polarity detection. In our setting, this step only takes subjective messages as input. It then determines whether the message contains positive or negative subjective statements. We first define this problem more crisply in Section 2.4.1. Related work is already referred to in Section 2.3.2. We present our RBEM algorithm to solve polarity detection in Section 2.4.2.

2.4.1 Problem Formulation

Polarity detection is a task that determines the polarity a message expresses. This can be a general case where the polarity can be one of *positive, negative, bipolar, neutral* where 'neutral' generally, but not necessarily, implies objectiveness. Consider for example the sentence *I think I just saw a car pass by* which

expresses a personal view but has no polarity. For our purposes, we regard a somewhat simpler problem by assuming that the *neutral* class always implies objectiveness. As subjectivity detection filters our objective messages from the succeeding process already, we need to assign to a given message m whether it is positive, negative or bipolar but we assume that either positive or negative sentiment always prevails when both occur. We thus only do this if a message actually does express a polarity, meaning that we only process messages labeled subjective by the subjectivity detection step. We approach this problem in a supervised manner where we use different models for different languages.

Taken our assumptions into account, the problem we solve is the following. Given n labeled messages $[m_1, .., m_n]$ of which the label can either be *positive* or *negative*, learn a model such that for an unlabeled subjective message t, we can determine whether it is positive or negative.

2.4.2 Our Approach

When we look at our problem formulation posed in Section 1.2, restated below, polarity detection solves the bold part concerning what the polarity is of the sentiment that is expressed.

Given a message written in an unknown language to be determined, does this message contain sentiment and if so, **what is the polarity of this sentiment***?*

To solve this problem of polarity detection, we propose an algorithm we call the *Rule-Based Emission Model* (RBEM) algorithm. The name of the algorithm indicates the concepts it carries as it uses rules to define an emissive model. Each entity in a message can emit positive or negative sentiment. The rules are defined on eight different types of patterns. The idea of using patterns arises from [Wilson et al., 2005] who label subjective expressions (patterns) in their training data. In their experiments however they limit themselves to matching against single-word expressions. The use of rules stems from a different domain. The Brill tagger [Brill, 1992] for POS-tagging uses rules. We borrow this ideology and apply it to polarity detection. The emission aspect of our RBEM algorithm is related to smoothing which is often applied in different machine learning settings for different purposes.

The rules used in the RBEM algorithm directly stem from eight different *pattern groups*, defined as follows. Note that the examples only list words but that a pattern can consist of any combination of words and POS-tags. This concept is further explained when we describe how to learn a model.

- **Positive** patterns are positive when taken out of context. English examples hereof are *good, well done*. We denote this group as *positive*.
- **Negative** patterns are negative when taken out of context. English examples hereof are *bad, terrible*. We denote this group as *negative*.
- **Amplifier** patterns strengthen polarity, either positive or negative. English examples hereof are *very much, a lot*. We denote this group as *amplifier*.
- **Attenuator** patterns weaken polarity, either positive or negative. English examples hereof are *a little, a tiny bit*. We denote this group as *attenuator*.
- **Right Flip** patterns flip the polarity of n entities to their right. English examples hereof are *not, no*. We denote this group as *rightflip*.
- **Left Flip** patterns flip the polarity of n entities to their left. English examples hereof are *but, however*. We denote this group as *leftflip*.
- **Continuator** patterns continue the emission of polarity. English examples hereof are *and, and also*. We denote this group as *continuator*.
- **Stop** patterns interrupt the emission of polarity. Stop patterns usually are punctuation signs such as a dot or an exclamation mark, expressing the general case that polarity does not cross sentence boundaries. We denote this group as *stop*.

The need for continuators and left flips has been indicated in earlier work by [Hatzivassiloglou and McKeown, 1997] who found that conjunctive words such as *and* usually connect adjectives of the same polarity whereas conjunctive words such as *but* usually connect words of opposing polarity. The need for positive and negative patterns is trivial. The remaining patterns stem from our own experience. Combining these eight pattern groups using some simple rules allows us to define an emissive model. We next describe how a model is constructed and then define how to classify previously unseen data. Definitions are presented when we first introduce them.

Learning a Model

Each message m of length n is represented as a list $m = [(w_1, t_1), .., (w_n, t_n)]$ of tuples of a word w_i with its respective POS-tag t_i. Upon such a message, patterns can be defined. A pattern is a list of tuples of words and POS-tags and are represented in the same fashion messages are. A pattern belongs to a single pattern group and hence we represent a pattern q as a tuple $q = (g, p)$ where g is the pattern group q belongs to and p is the list of entities comprising the actual pattern. In general, each element (w'_i, t'_i) of a pattern p consists of a word w'_i which is precisely defined and a POS-tag t'_i which is also precisely defined. As an exception, elements of p may contain wildcards instead. We consider three types of wildcards.

- **Word wildcards** – In this case we only consider t'_i. w'_i can be any arbitrary word. This type of wildcards can occur at any position in p. Such an element is represented as $(\_, t'_i)$.
- **Single-position wildcards** – In this case a single entity can be any arbitrary combination of a single word and a single POS-tag. This type of wildcards can occur at any position in p. Such an element is represented as $(\_, \_)$.
- **Multi-position wildcards** – In this case any arbitrary combination of word and POS-tag pairs of any arbitrary length matches the pattern. This type of wildcards can only occur in between two elements that are not multi-position wildcards nor single-position wildcards as co-occurrence with either of these two wildcards yields another multi-position wildcard. Such an element is represented as $(*, *)$.

Our model now simply consists a set of patterns per pattern group, represented as the set *Model* containing tuples of groups and patterns (g, p). Every pattern except for the positive and negative patterns adhere to an action radius $\mathcal{E}$. This action radius is also part of our model. In this work, we set $\mathcal{E} = 4$. The choice is not arbitrary but again inspired by the work of [Wilson et al., 2005] who mention using local negation with a frame of four words yielded the best results among other frame lengths.

Classification

When classifying previously unseen data, we perform two steps. First we collect all patterns in our model that match our sentence. Second, we apply a rule associated with each pattern group for each pattern present in our message. We first describe how patterns present in our model are matched against our message and next describe the rules applied to these patterns.

Pattern Matching

Each pattern $q = (g, p) \in$ *Model* is matched against our message $h = [(w_1, t_1), .., (w_n, t_n)]$ where $p = [(v_1, s_1), .., (v_m, s_m)]$. We consider each tuple (w_i, t_i) and evaluate $(v_1, s_1) =_{match} (w_i, t_i)$ where $=_{match}$ is defined as follows.

$$(v_j, s_j) =_{match} (w_i, t_i) \equiv \begin{cases} \text{true} & (1) \\ \quad \text{if } j > m, \text{ define } end \leftarrow i & \\ \text{false} & (2) \\ \quad \text{if } i > n & \\ v_j = w_i \wedge s_j = t_i \wedge (v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1}) & (3) \\ \quad \text{if } v_i \neq \_ \wedge v_i \neq * \wedge j \leq m \wedge j \leq n & \\ s_j = t_i \wedge (v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1}) & (4) \\ \quad \text{if } v_i = \_ \wedge s_i \neq \_ \wedge j \leq m \wedge j \leq n & \\ (v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1}) & (5) \\ \quad \text{if } v_i = \_ \wedge s_i = \_ \wedge j \leq m \wedge j \leq n & \\ (v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1}) \vee (v_j, s_j) =_{match} (w_{i+1}, t_{i+1}) & (6) \\ \quad \text{if } v_i = * \wedge j \leq m \wedge j \leq n & \end{cases}$$

Note that in the definition of $=_{match}$, cases (4), (5) and (6) correspond to the three different types of wildcards. Moreover, in the evaluation of the first disjunction of (6), $(v_{j+1}, s_{j+1}) =_{match} (w_{i+1}, t_{i+1})$, it must hold that $v_{j+1} \neq * \wedge s_{j+1} \neq * \wedge \neg(v_{j+1} = \_ \wedge s_{j+1} = \_)$ due to the restriction we put on the occurrence of multi-position wildcards.

We match all patterns of all groups against every possible element (w_i, t_i) of m. While doing this, we need keep track of two positions used if a pattern matches; the start position of the match in m and the end position of the match in m. The starting position is i whereas the end position is *end* which is assigned a value in case (1) of $=_{match}$, implying a match between the pattern and the message. We thus get a set of matching patterns containing a start position, an end position and a pattern.

$$matchedPatterns = \{(start, end, (g, [(v_1, s_1), .., (v_n, s_n)])) \mid (v_1, s_1) =_{match} (w_{start}, t_{start})\}$$

Elements of *matchedPatterns* may subsume each other. Subsumption in this sense is defined as follows, where we say that q_1 subsumes q_2.

$$\exists_{(s_1,e_1,q_1),(s_2,e_2,q_2) \in matchedPatterns} : (s_1 = s_2 \wedge e_1 > e_2) \wedge q_1 \neq q_2$$

All patterns that are subsumed by some other pattern are removed. Note that coinciding patterns, having the same start position as well as the same end position, are not removed but as we deal with sets, such coinciding patterns must be of different pattern groups. The resulting set *maxPatterns* only contains maximal patterns and is defined as follows.

$$maxPatterns = \{(s, e, q) | (s, e, q) \in matchedPatterns \wedge \\ \neg(\exists_{(s', e', q') \in matchedPatterns} : (s = s' \wedge e' > e) \wedge q \neq q')\}$$

Rule Application
After having collected all maximal patterns, we can apply the heuristic rules for each different pattern group. The rules formally work out the motivation for the presence of each pattern group. The order in which the rules are applied is crucial and so is the role of the action radius $\mathcal{E}$. We outline each of the rules in the order in which they are to be applied. We assume we are given a message m and a model $(Model, \mathcal{E})$ on which *maxPatterns* is defined. Every element $e_i = (w_i, t_i) \in m$ has a certain emission value $em(e_i)$ which initially is set to 0 for all $e_i \in m$.

Rule 1. Setting Stops – This rule sets emission boundaries in our message m. It uses all left flip and stop patterns and sets a stop at the starting position of such a pattern. We thus get a set of stops.

$$stops = \{s | (s, f, leftflip) \in maxPatterns \vee (s, f, stop) \in maxPatterns\}$$

Rule 2. Removing Stops – Stops set in the previous step can be removed by continuator patterns. This however, only happens to the left of a continuator pattern. We thus remove all stops that occur closest to the left of a continuator pattern, taking $\mathcal{E}$ into account.

$$stops = stops \setminus \{t | t \in stops \wedge (\exists_{(s,f,continuator) \in maxPatterns} : t \leq s \wedge s - t < \mathcal{E} \wedge \neg(\exists_{t' \in stops} : t < t' \leq s))\}$$

Rule 3. Positive Sentiment Emission – A positive pattern can emit positive sentiment among elements of m. The strength of the emission decays over distance and hence we need a decaying function. We use e^{-x} as decaying function, where x is the distance between the positive pattern and an element of m. As center for the emission, we take the floor of the center of the pattern in m, computed by taking the center of start and end position. We also need to take all stops into account. For each positive pattern, we update the emission values $em(e_i)$ as follows.

$$\forall_{(s,f,positive)\in maxPatterns} : c = \lfloor \frac{s+f}{2} \rfloor \wedge (\forall_{e_i \in m} : \\ \neg(\exists_{t\in stops} : c \geq i \Rightarrow i \leq t \leq c \vee i \geq c \Rightarrow c \leq t \leq i) \Leftrightarrow \\ em(e_i) = em(e_i) + e^{-|c-i|})$$

Rule 4. Negative Sentiment Emission – Negative patterns are dealt with in the same way positive patterns are. The only difference is that our decaying function is now negative, yielding $-e^{-x}$. The updating of emission values happens in the same manner.

$$\forall_{(s,f,negative)\in maxPatterns} : c = \lfloor \frac{s+f}{2} \rfloor \wedge \\ (\forall_{e_i \in m} : \neg(\exists_{t\in stops} : c \geq i \Rightarrow i \leq t \leq c \vee i \geq c \Rightarrow c \leq t \leq i) \Leftrightarrow \\ em(e_i) = em(e_i) + -e^{-|c-i|})$$

Rule 5. Amplifying Sentiment – Amplifier patterns amplify sentiment emitted either by positive or negative patterns. Similar to the decaying function used for positive and negative patterns, amplification diminishes over distance. Moreover, since entities may already emit sentiment, we use a multiplicative function instead of an additive function. The function we use is $1 + e^{-x}$ where x is the distance. In contrast to positive and negative patterns, amplifiers adhere to the action radius $\mathcal{E}$. The emission values are updated as follows.

$$\forall_{(s,f,amplifier)\in maxPatterns} : c = \lfloor \frac{s+f}{2} \rfloor \wedge \\ (\forall_{e_i \in m} : (\neg(\exists_{t\in stops} : c \geq i \Rightarrow i \leq t \leq c \vee i \geq c \Rightarrow c \leq t \leq i) \wedge 0 < |c-i| < \mathcal{E}) \Leftrightarrow \\ em(e_i) = em(e_i) \cdot (1 + e^{-|c-i|}))$$

Rule 6. Attenuating Sentiment – Attenuator patterns perform the reverse of amplifier patterns and weaken sentiment. To do so, instead of using $1 + e^{-x}$, we use $1 - e^{-x}$.

$$\forall_{(s,f,amplifier)\in maxPatterns} : c = \lfloor \frac{s+f}{2} \rfloor \wedge \\ (\forall_{e_i \in m} : (\neg(\exists_{t\in stops} : c \geq i \Leftrightarrow i \leq t \leq c \vee i \geq c \Leftrightarrow c \leq t \leq i) \wedge 0 < |c-i| < \mathcal{E}) \Leftrightarrow \\ em(e_i) = em(e_i) \cdot (1 - e^{-|c-i|}))$$

Note the $0 < |c-i| < \mathcal{E}$ clause. This constraint dictates that x is at least 1 in $1 - e^{-x}$, thus avoiding the case that we multiply by 0 and hence completely remove emission values.

Rule 7. Right Flipping Sentiment – Right flip patterns simply flip the emission of sentiment to their right as follows. If there is a stop at the exact center of our right flip, we disregard it.

$$\forall_{(s,f,rightflip)\in maxPatterns} : c = \lfloor \frac{s+f}{2} \rfloor \wedge (\forall_{e_i \in m} : (\neg(\exists_{t\in stops} : \\ c < t \leq i) \wedge |c-i| < \mathcal{E}) \Leftrightarrow em(e_i) = -em(e_i))$$

Rule 8. Left Flipping Sentiment – Left flip patterns mirror the effect of right flip patterns.

$$\forall_{(s,f,leftflip)\in maxPatterns} : c = \lfloor \frac{s+f}{2} \rfloor \wedge (\forall_{e_i \in m} : (\neg(\exists_{t\in stops} : \\ i \leq t < c) \wedge |c-i| < \mathcal{E}) \Leftrightarrow em(e_i) = -em(e_i))$$

Once the above rules have been applied in the order given, every element e_i of m has an emission value $em(e_i)$. The final polarity of the message is defined by the sum of all emission values for all elements of

m.

$$polarity = \sum_{i=1}^{n} em(e_i)$$

Straightforwardly, we say that m is *positive* if and only if $polarity > 0$. Likewise, we say that m is *negative* if and only if $polarity < 0$. Whenever $polarity = 0$, we do not know the polarity of m, either due to insufficient patterns present in our model or due to the absence of patterns in m.

Alternatively to using 0 as the decision boundary, one could use some threshold t. We then say m is *positive* if and only if $polarity > t$, *negative* if and only if $polarity < -t$ and unknown otherwise. We however do not consider using such a threshold.

Example 3
We demonstrate the operation of the RBEM algorithm through an unrealistic example purely designated for illustrative purposes. Consider having the model shown in Table 1. The model contains two patterns of which one contains a wildcard element. We now consider the following message *I like the green car more than the red car*. Represented as a list of token and POS-tag pairs we get; $[(w_1, t_1), .., (w_{10}, t_{10})] = [$(I,PP), (like,VVP), (the,DT), (green,JJ), (car,NN), (more,RBR), (than,IN), (the,DT), (red,JJ), (car,NN)$]$.

Constructing the set *matchedPatterns* requires us to evaluate $=_{match}$. As both patterns in our model match our message exactly once, following the definition of $=_{match}$ results in the following set.

$$matchedPatterns = \{(1, 2, (Positive, [(\_, NP), (like, VVZ)])), (6, 6, (Amplifier, [(more, RBR)]))\}$$

The *matchedPatterns* set thus contains two elements. As these two elements do not overlap, we do not need to filter out subsumed patterns. Our set *maxPatterns* is thus equal to *matchedPatterns*.

We can next apply the rules consecutively. We start with Rule 1 but as *maxPatterns* does not contain any stop patterns or left flip patterns, we do not need to set any stops. We also do not have any continuator patterns and hence we take no action with Rule 2. Rule 3 next requires us to emit positive sentiment for all positive patterns. As we have one such pattern, we need to emit positive sentiment once. The center c of our pattern is 1.5, since we take the floor hereof, we get $c = 1$. This yields the emission rates shown in the *Emission value 1* column of Table 2 which are exponential values of e. We skip Rule 4 as there are no negative patterns. Rule 5 next requires us to strengthen emissions as we have an amplifier pattern in our *maxPatterns* set. Multiplying all emission rates with $1 + e^{-|c-i|}$, having a center value of $c = 5$, yields the emission values shown in the *Emission value 2* column of Table 2. Note that for amplifier patterns, we need to adhere to our emission range $\mathcal{E} = 4$. As we have no more patterns in *maxPatterns*, the emission values shown in the *Emission value 2* column of Table 2 are the resulting emission values.

The label is now determined by looking at the sign of the sum of all emission values. As all emission values are positive, the sign is also positive, indicating that the message's polarity is positive.

Table 1: An example model for the RBEM algorithm.

Pattern	Type
[(_,NP),(like,VVZ)]	Positive
[(more,RBR)]	Amplifier

Table 2: Emission values for the example.

Index	Message element	Initially	Emission value 1	Emission value 2
1	(i,PP)	0	$e^{0} = 1.000$	1.000
2	(like,VVP)	0	$e^{-1} = 0.368$	0.368
3	(the,DT)	0	$e^{-2} = 0.135$	0.142
4	(green,JJ)	0	$e^{-3} = 0.050$	0.057
5	(car,NN)	0	$e^{-4} = 0.018$	0.025
6	(more,RBR)	0	$e^{-5} = 0.007$	0.013
7	(than,IN)	0	$e^{-6} = 0.002$	0.003
8	(the,DT)	0	$e^{-7} = 0.001$	0.001
9	(red,JJ)	0	$e^{-8} = 0.000$	0.000
10	(car,NN)	0	$e^{-9} = 0.000$	0.000

3 Experimental evaluation

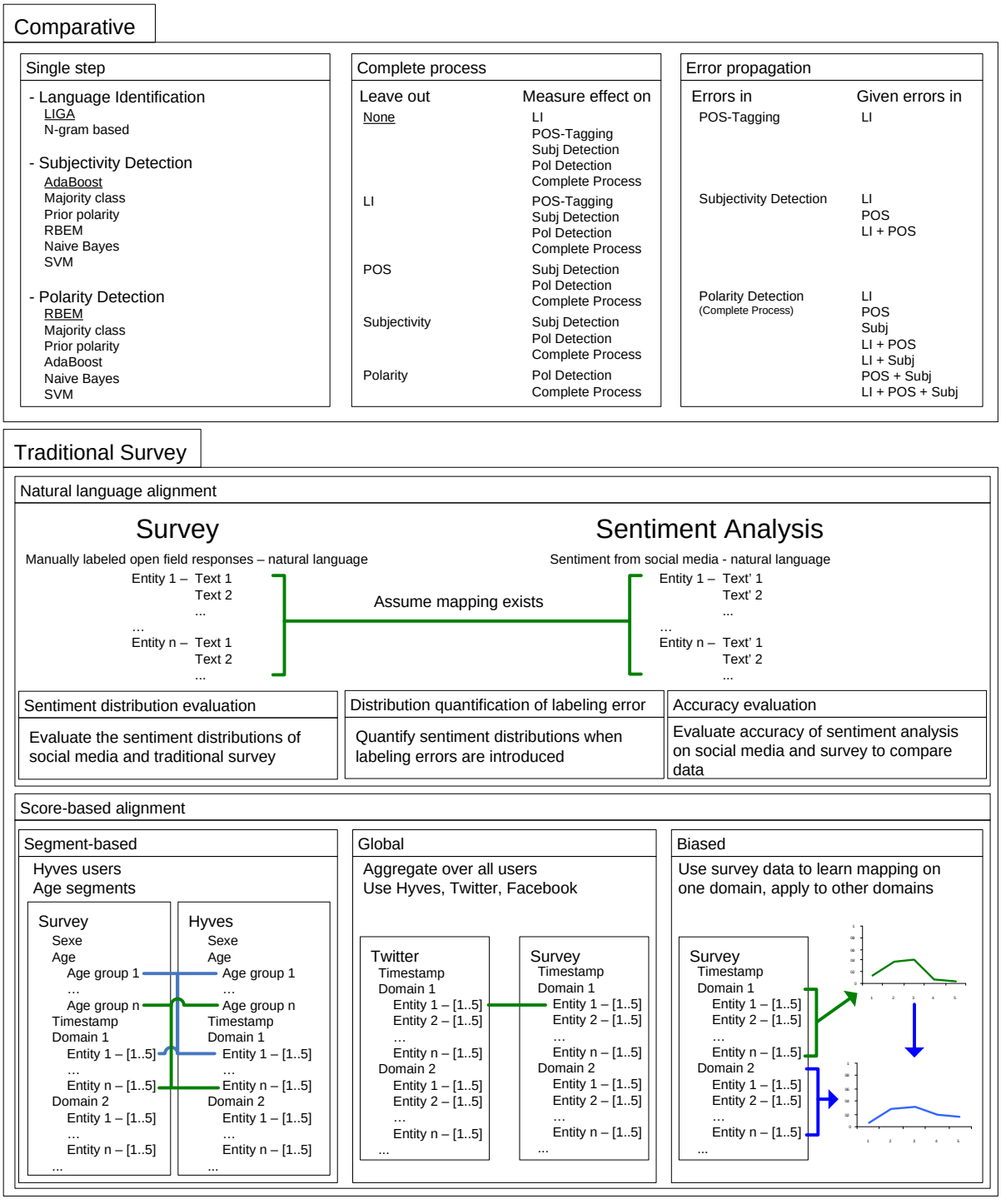


Figure 5: Experiment setup. Comparative experiments look at our sentiment analysis solution itself whereas for the traditional survey experiments we analyze the (dis)similarity of our sentiment analysis approach and a traditional survey.

Figure 5 shows all experiments we conduct. The experiments are divided into two main categories; *comparative* experiments and *traditional survey* experiments. The former type of experiments are concerned with comparing our sentiment analysis approach against other approaches, either taking the complete process or a single step into account. In the latter type of experiments we show the difficulties that arise when aligning our sentiment analysis' results against that of a traditional survey, either by globally aggregating over the results or by taking specific segments into account.

To demonstrate the actual multilingual aspect of our approach, all comparative experiments are performed on multilingual data encompassing Dutch and English messages. We only use these two languages as we have sufficient knowledge on them and can hence evaluate the experiments without jeopardizing the results and still demonstrate the multilingual part. If one would like to experiment with more languages, the same process we show with just English and Dutch can be performed but then incorporating those other languages as well. With incorporating more languages, special attention needs to be paid to polarity detection though as the model's patterns need to be added manually for the RBEM algorithm which is not a trivial nor easy task. For the other steps, less difficulties arise as less human supervision is required.

To argument why we perform the experiments presented in Figure 5, we first give a motivation and a brief outline for each of the experiments in Section 3.1. When performing the experiments we present, special attention needs to be paid to prevent us from overfitting our models or biasing our results. When we present the motivation for each experiment, we point out possible dangers of overfitting or how biased results may arise. We perform our experiments using four main datasets. These datasets are described in Section 3.2. The actual experiments are more elaborately described in Section 3.3 and 3.4 where we also present and analyze the corresponding results.

3.1 Motivation & Goals

The main goals of our experimental study are to show that 1) the choice of the algorithm used for each of the four steps is justified and that 2) the use of a four-step approach is justified, meaning that it is expected to perform better than more generic approaches to sentiment analysis. Besides these goals, we also study the importance of step. We discuss these experiments in more detail in Section 3.1.1.

For the second line of experiments we conduct, we study how automated sentiment analysis can be aligned to traditional survey data. The experiments that deal with this line are described in more detail in Section 3.1.2.

3.1.1 Comparative Experiments

Single Step
The single step experiments compare our chosen approach for three out of the four steps, being LIGA for language identification, AdaBoost for subjectivity detection and RBEM for polarity detection, against other baselines and competitor approaches. Through these experiments we show that the locally chosen solutions perform better than other competitor approaches and yield a substantial increase over the baseline approaches. The POS-tagging step is not experimented with here as this is experimented with in [Schmid, 1994]. We do not attempt to reproduce or verify the results presented in [Schmid, 1994] but simply rely on them and assume similar performance in our context. The approaches we use and the baselines and competitor approaches we perform against are listed in Figure 5.

When performing these single step experiments, we need to be careful to make a fair comparison. If we use training data upon which we specifically optimize parameters of our chosen approaches, we cannot just compare to another approach which is not optimized on our data. We can thus either choose to optimize all approaches on our data or not to optimize at all and use different parameter settings to show the influence of parameters without the intention to optimize. As optimizing all approaches is a tedious task, we choose to use different parameter settings for each approach and compare to all different settings that we used.

Briefly summarized, the single step experiments achieve **Goal 1** stated below.

Goal 1. Collect empirical evidence supporting (or rejecting) the choice of the solutions used in each step of our four-step sentiment analysis approach is justified, i.e. they perform better than other algorithms serving the same purpose.

Complete Process

Even though we use approaches that may show to perform better than other baselines or competitors, it may be that one or more steps actually only introduce propagation errors into other steps. For example, if our language identification is only 80% accurate, the POS-tagger will apply a wrong model in 20% of all cases whereas a more generic approach not taking the language into account may do better. To this end we investigate how each step benefits the complete process as well as solving each of its succeeding steps. We do so by first performing the four step process after which we know the performance of each step and combination of steps in what we define as the specialized standard. We choose to use this term as the four step process allows for more specialized approaches in a succeeding step given the knowledge from preceding steps. We next leave out each step exactly once, yielding a more general approach, and compare the performance of each of the remaining steps with the specialized standard. This gives us a complete overview of the impact the left-out step has on the complete process. Note that leaving out a single step only influences succeeding steps, which is reflected in Figure 5.

Leaving out a single step means we lose some information in succeeding steps and hence need to apply a more generic approach for succeeding steps. We can do this in three different ways.

- Still use the solution we of our four-step process for a succeeding step but use a heuristic to overcome not having the information of previous steps. An example hereof is when we look at language identification and POS-tagging. POS-tags differ per language and hence the models for our POS-tagger are language specific. We can choose the right model because we know the language of a message from our language identification. If we now leave out language identification, we can no longer choose the right model for our POS-tagger. What we can do however is apply all models that are present and choose the most likely one. This is in fact an encapsulated way of performing language identification through a POS-tagger.
- We can replace our approach for a step with a more generic approach that does not rely on previously collected data. Consider for example performing subjectivity detection with AdaBoost using language-specific models based on POS-tags. If we now leave out POS-tagging, we can no longer use the POS-tags in our models as we do not have this information. What we can do however is use tokens as features instead and then either decide whether to still use AdaBoost or perhaps another approach such as Naive Bayes.
- When we leave out a step, it may be that we can no longer straightforwardly give an answer to the problem formulation given in Section 1.2. With our four-step approach, this situation only occurs when we leave out either subjectivity detection or polarity detection. We in fact require a three-way classification as we have three possible answers to our problem: *positive, negative, objective*. Our subjectivity detection performs a two-way classification deciding whether a message is either objective or subjective in which case it is one of positive and negative but it does not determine which of the two. Polarity detection then performs a two-way classification determining whether the message is positive or negative. If we leave either of subjectivity detection or polarity detection out, we can no longer perform this three-way classification. We thus need to adjust our approach by coping with three classes. When we leave out subjectivity detection, this means that our RBEM algorithm for polarity detection not only needs to determine whether a message is positive or negative but first whether it is subjective or objective. When we leave out polarity detection, this means that our AdaBoost algorithm not only needs to determine whether a message is subjective or objective but also – if it is subjective – whether it is positive or negative.

For the complete process experiments we thus compare a more specific setting, where knowledge from previous steps is present, with a more generic setting where some knowledge is missing. If we choose to use each of the single steps of our four-step process because they show to outperform other approaches on validation data, we can no longer use this validation data to compare our specific setting against a more generic setting as the specific setting as this would bias our results because we used the validation data constructively to define our specific setting.

The goal achieved by these complete process experiments is **Goal 2** stated below.

Goal 2. Determine the benefit of each single step for solving succeeding steps and the complete process.

Error Propagation
Even when we know that our solutions perform better than other solutions and that we have shown the need of each step, it is still important to know how the error of one step propagates to succeeding steps. This gives insights into where we should try to improve and where it is little use. It will also tell us which steps are more important to have as accurate as possible. We also relate the error propagation to generic baseline approaches that perform sentiment analysis in a single step. This will show what the thresholds are when using our cascaded four-step approach is no longer beneficial as opposed to using a generic single-step approach.

To analyze the propagation of errors, we construct a highly accurate dataset of which we are confident that all labels are correct as input for a single step. This dataset is fully and manually labeled on language, POS-tags, subjectivity and polarity. As the output of each step serves as input for the succeeding step, we can leave out a first step and use the highly accurate dataset as input instead. Since the dataset contains little to no errors, we can explicitly introduce errors and quantify how these errors propagate. Note that as we talk about propagation, we can only analyze the effect errors have on succeeding steps as is shown in Figure 5.

With analyzing the error propagation, we need to pay the same attention to not biasing our results as we need for the complete process experiments. We cannot use the validation data on which we evaluated and constructed our four-step process when analyzing the propagation of errors as would bias our results since our four-step approach may have been optimized to perform well – and hence make less errors – on the validation data.

Through analyzing the error propagation we achieve **Goal 3** stated below.

Goal 3. Quantify the error propagation of each step on every succeeding step.

3.1.2 Traditional Survey Experiments

For the traditional survey experiments we illustrate the difficulty of making an alignment between the results of our automated sentiment analysis against the results of a traditional survey typically used in solving marketing questions. We investigate an alignment using natural language for our sentiment analysis process against using natural language in the traditional survey. We additionally study making mappings using a score-based rating for our traditional survey instead. With these experiments we achieve **Goal 4** stated below.

Goal 4. Investigate how sentiment analysis relates to addressing marketing questions. More specifically, we want to know how sentiment analysis maps to traditional surveying.

In all experiments we use a traditional survey conducted amongst Dutch citizens. To make a fair analysis, we hence only use Dutch messages from our sentiment analysis. Moreover, we know for each survey respondent which social media he or she uses and hence we only take those respondents into account that use a particular social medium against our sentiment analysis on that social medium in our analysis.

The traditional survey in constructed to ask for an opinion on entities divided into four different domains. This allows for a per-entity mapping as well as a per-domain mapping. For each entity, a respondent is asked to give a score-based rating and give a 'comment' where sentiment can be expressed in natural language. We investigate making an alignment using the natural language comments and additionally look at using the score-based ratings. We next describe how we do this.

Natural Language Alignment
We manually label the responses in the comment fields of the traditional survey to be either one of *positive, negative, objective*. As labeling all such comment fields is too tedious, we use samples instead. We also manually label messages crawled from social media to be either one of *positive, negative, objective*. These messages are in Dutch as our survey is as well and the messages apply to the same entities our

survey does. We assume that our labeling on both datasets is ground truth and that a mapping between the two exists. We do not explicitly know the mapping but due to its existence, metrics defined on the two datasets can be compared.

We compare the distributions in sentiment of both datasets as a metric. We define a reference point by using the 100% accurate datasets which is hence assumed to be ground truth. We next compare the sentiment distributions that result from running our sentiment analysis on the data from social media to our reference point. We can then evaluate whether or not our sentiment analysis is accurate enough to establish the mapping we assumed to exist.

We also study the influence error rates have on the sentiment distributions. We deliberately introduce errors in our ground truth dataset based on social media messages. We do this with varying amounts of errors and investigate the resulting sentiment distributions by comparing them to the sentiment distributions of the traditional survey as well as the sentiment distributions resulting after applying our sentiment analysis on the social media data.

To compare the two data sources – the social media ground truth data and the traditional survey ground truth data – we use accuracy as a metric. We perform our sentiment analysis on both data sets and compare the accuracy to evaluate the similarity of the two data sets.

Score-Based Alignment
For the score-based alignment, we use the score-based ratings respondents provide on each entity. We compare possible mappings on the segment-level, the global level and investigate what it implies when we train our mapping on the survey data, thus biasing our results.

The segment-based mappings take segment into account. In our specific case, we use a person's age to segment upon. As Hyves is the only social medium that allows us to obtain age information from its users, we only use this social medium in our segment-based analysis. For the global mappings we do not segment in any way and hence do not regard age. This means that we can drop the constraint that we need to be able to extract age information from our social medium and hence we can use all social media.

There is an inherent problem with trying to align our sentiment analysis' results and those of a traditional survey as we deal with two inherently different worlds, opinions given by survey respondents against opinions given by social media users. An example of the inherent difference is shown in Table 3 where possible answers of the survey and for sentiment analysis are shown. We ideally want to use domain knowledge from the way these two different worlds express their opinion to determine a correct mapping between the two worlds. This difference in the two worlds would be a study on its own and we do not attempt to study this in our work. Using domain knowledge biases our results as we use our mapping material to construct the mapping itself. Our results thus become biased. We investigate what the implications are of having such biased results, especially how much this helps over not using prior knowledge.

Table 3: An illustrative example of survey answers and sentiment analysis (SA) answers. For the survey an answer is a number showing the rating of the entity. With sentiment analysis, we only have natural language.

Entity	Survey answers	SA answers
e_1	1,1,3,2,5,1,3,..	I like e_1, don't think e_1 is good,
e_2	4,5,4,3,3,2,4,..	e_2 is my thing, just bought e_2,..
e_n	3,3,3,5,1,2,3,..	what is wrong with e_n, I love e_n,..

3.2 Datasets

We presented four goals for our experiments in Section 3.1. To achieve these goals we need to train and evaluate our approaches on suitable data. In our case, this is not a trivial task as we cannot just use one single dataset in each setting since the settings are different in nature. Figure 6 shows the four main different datasets we use. The relation between an experiment and a dataset is shown in Figure 7.

In all of the experiments, one or more of the four steps of our complete process are used. As each of the

steps need to be trained before we can evaluate it, we also require training data. Since each step serves as a different classification task, the training data used for each step also differs. The mapping from a single step to the training data used to learn a model for that step is shown in Table 4[7].

We describe the construction and purpose of each dataset in Sections 3.2.1 through 3.2.5. With each dataset we motivate why the dataset is required by directly linking it to one of the goals presented in Section 3.1. All datasets are available at `http://www.win.tue.nl/~mpechen/projects/smm/`.

Table 4: The training data used for each step.

Step	Training data
Language Identification	Separate training data described in Section 3.3.1
POS-tagging	Standard models of the TreeTagger website
Subjectivity Detection	Training set shown in Figure 6
Polarity Detection	Training set shown in Figure 6

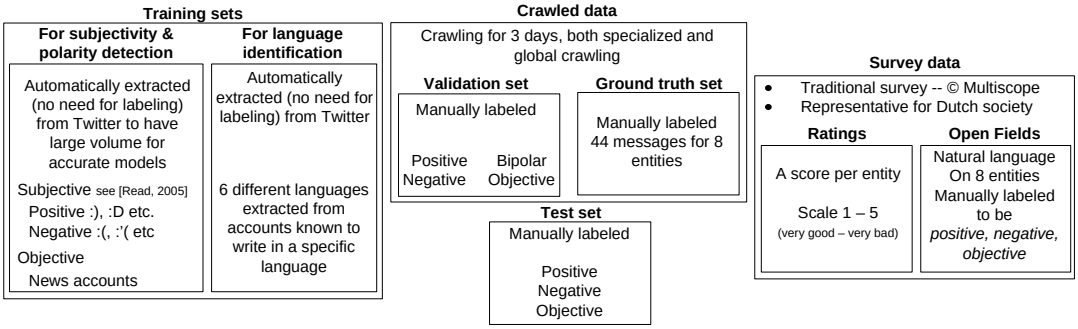


Figure 6: The datasets used in the experiments. For the training data we use smilies and news instances as noisy labels for subjective and objective sentiment to easily obtain a lot of data. The training set for language identification more extensive and described in Section 3.3.1. The validation set as well as the ground truth set are taken out of the crawled data and are manually labeled. The test set is a different dataset also manually labeled. The survey data contains survey results representative for Dutch society, being either score-based ratings or open field containing natural language.

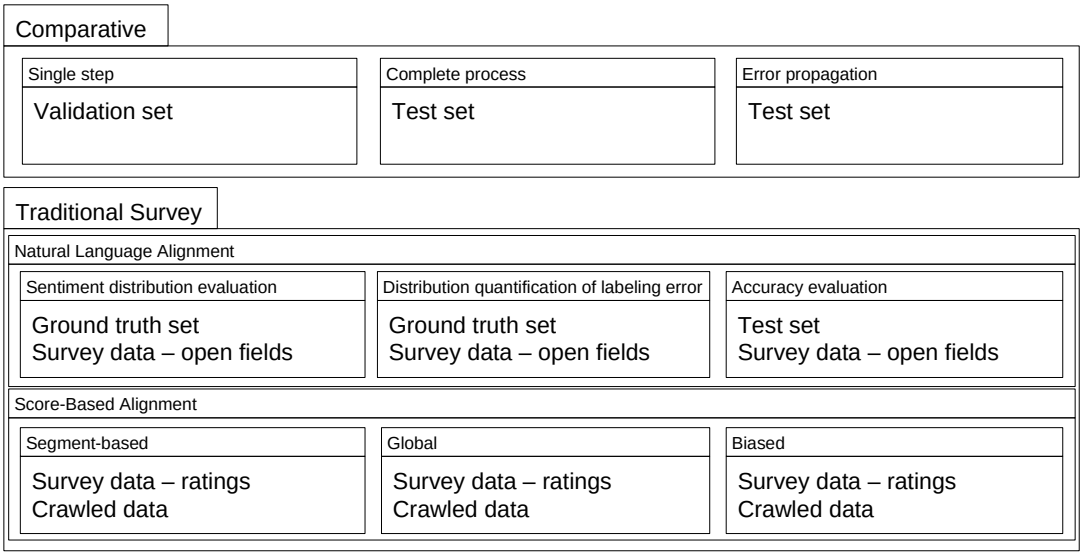


Figure 7: The mapping between experiments and datasets.

3.2.1 Survey Data & Crawled Data

Goal 4 already mentions mapping to a traditional survey. This inherently requires us to create such a survey. This is the survey data shown in Figure 6. We cannot align such a traditional survey against any

[7] The website of the TreeTagger is located at http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

randomly chosen data, the alignment needs to be as unbiased as possible. We therefore require a dataset on the same domain the traditional survey is, collected in the same timespan. This dataset is the crawled data shown in Figure 6. The models of our sentiment analysis are trained as per Table 4.

The survey data is obtained by surveying a representative sample of Dutch society, as defined by the CBS[8]. The survey – created by Multiscope for this study – that was filled in by respondents is presented in Appendix D. Respondents are asked to rate sixteen different entities which are grouped together in four different domains as per Table 7. The rating of such an entity is on a scale from 1 (very good) to 5 (very bad). Additionally, the respondent can mark having no opinion on the entity in which case we leave this respondent's entry out for that particular entity. Before any entities are marked, the respondent is asked what social media (any of Hyves, Twitter, Facebook) he or she uses, allowing us to create a mapping per social medium. The survey data results directly from the respondents' answers to these questions. We additionally know each respondent's age, allowing for a segment-based alignment.

If we want to align the results of the survey with those of our sentiment analysis, we need to be sure that the results are based on similar data. To this end, we crawl all social media specifically for the entities presented in Table 7 by querying the social media for all messages containing one of our entities. It may occur that a message is about one of our entities but because of for example a misspelling we cannot find it by querying the social media for it. To still obtain these messages, we additionally collect all messages that is publicly available on all social media. As the survey was set out for a period of three days and since the crawling took place in parallel, crawling also stopped after three days. A part of this data is used to create the validation set described in Section 3.2.4 as shown in Figure 6. The remainder of the crawled data is completely classified by our four-step process and is used to create a mapping onto the survey's data, resulting in the crawled data shown in Figure 6.

Since the survey is only answered by Dutch citizens, we disregard all messages not written in Dutch as classified by our language identification algorithm. Note that in addition to just collecting messages, we also collect age information in case of the Hyves social network. Using the age feature, we can thus perform a more fine-grained analysis.

We will use the open field data of the traditional survey as a ground truth. In order to use these open fields as a ground truth, we need to manually inspect the data and determine for each open field answer whether it contains an opinion and if so, whether it is positive or negative. The survey contains responses of a representative sample of Dutch citizens of 1429 respondents. Out of these 1429 responses, exactly 919 filled in at least one of the open fields on at least one of the sixteen entities presented in Table 7.

Manually inspecting all of the open field responses is not possible within a reasonable timespan. We therefore only select a sample of those 919 respondents. Since the complete survey has been filled in by a representative sample however, we will use its distribution in age to create our sample out of the 919 respondents that we will manually inspect and assign either one of *objective, positive, negative* to each open field response. As not all of the 919 respondents who gave their opinion in at least one open field filled in all of the open fields, we will construct a representative sample for each entity separately to make sure that our sample is representative per entity. Note that this means we do not have a representative sample with respect to the number of opinions given but as this is not what we want to investigate, this is not a problem. We create two datasets from the open field responses.

The complete distribution of the number of respondents that filled in an open field belonging to an entity is shown in Table 5. For each entity we select exactly 60 open field answers given by respondents sampled to be representative for Dutch society, following the distribution shown in Figure 16(a). We use only 60 open field responses per entity – resulting in a total of 960 responses as we have 16 different entities – as we need to manually label each response which is a tedious task. This results in using the number of responses per age group as shown in Table 6 for each entity. To obtain the amount of responses shown in Table 6, we uniformly sample from each age group per entity, taking our legitimacy constraint into account. This gives us a sample dataset that is still representative for Dutch society.

We have already pointed out that on Hyves, only the [18, 29] age group is properly represented. To this end, it makes sense to perform a segment-based analysis mainly focusing on this age group. The number

[8]Centraal Bureau voor de Statistiek - http://www.cbs.nl/en-GB/

of open field responses per entity for this age group are much too low in our representative sample though. We thus create an additional ground truth sample using the same methodology consisting of only 50 open field responses per entity in the [18, 29] age group and no open field responses for the other age groups. We use the number 50 since there is one entity (*Uri Rosenthal*) that has only 57 responses which we round down to the multiples of ten. We thus have the following two ground truth samples.

Sample 1. Representative sample containing for each age group the number of open field responses shown in Table 6 to be representative for Dutch society.

Sample 2. Low-age group sample containing exactly 50 open field responses per entity only for the [18, 29] age group, and nothing else.

The labeling of each of the open field responses is done in a stand-alone manner. This means that whenever an entity is rated positively or negatively but its corresponding open field response is neutral when not looking at the rating, we label the message as being neutral. We do this since it can only be determined positive or negative if we use rating information which we want to avoid as we then again not only use natural language. We moreover only use an open field answer if it contains legitimate text that stands on its own (i.e. an open field answer such as – or *idem* is not used).

Once we have our labeled open field data, we know for each natural language response whether it is positive, negative or objective. As this labeling was performed manually combined with the fact that the samples are representative for Dutch society, we regard it as a ground truth labeling upon which we can evaluate our sentiment analysis process. Since we now have the same labels for our traditional survey as we have for our sentiment analysis, the mapping problems identified in Section 3.4.1 are no longer present.

Table 5: The number of open field responses per entity if the entity was actually rated.

Entity	Responses	Entity	Responses
Manchester United	376	Playstation 3	223
FC Barcelona	392	Wii	381
AC Milan	365	Xbox	209
Sloggi	504	Wilders	622
Bjorn Borg	433	Rosenthal	449
Nike	486	Cohen	570
Mexx	416	Rutte	574
Esprit	454	Sap	509

Table 6: The number responses per age group.

Age Group	Responses
[18, 29]	12
[30, 39]	10
[40, 49]	11
[50, 59]	13
[60, 84]	15

Table 7: The entities a respondent is asked to give an opinion on in the survey.

Domain	Entity
Soccer	Manchester United, Fc Barcelona
	AC Milan
Game consoles	Nintendo Wii, Sony Playstation 3
	Microsoft Xbox 360
(Clothing) Brands	Nike, Mexx
	Sloggi, Björn Borg
	Esprit
(Dutch) Politicians	Geert Wilders, Mark Rutte
	Job Cohen, Uri Rosenthal
	Jolande Sap

3.2.2 Ground Truth Set

The open field data set described in Section 3.2.1, resulting from the traditional survey's data, is to be aligned against similar data from resulting from social media to achieve **Goal 4**. This similar data is our ground truth set. Our ground truth set is taken from our crawled data. As our survey asks for opinions on different entities, we specifically only include social media messages on those entities in our ground truth set to make a fair alignment.

We use sixteen entities in total in our traditional survey. Our crawled data contains tens of thousands of messages on these entities. As we need to manually label all messages to indeed obtain a ground truth, labeling all messages is too tedious. We therefore use exactly 44 messages for eight entities; *FC Barcelona, Manchester United, Playstation 3, Wii, Esprit, Nike, Job Cohen, Geert Wilders*. We thus use two entities per domain. The choice for the specific entities and the choice of using exactly 44 messages have no rationale.

As our survey was held among Dutch citizens, each of the messages present in our ground truth set is also in Dutch. We label each of these messages to be either *positive, negative* or *objective*. Note that the only information we have for each of these messages is the message itself as the message may originate from any of the social media we crawl. We can thus not segment on for example age as for Twitter and Facebook we do not have age information.

3.2.3 Training Sets

As all algorithms used in our sentiment analysis are supervised, we first need to train a model on labeled training data. As shown in Figure 6, we have two training sets. The models for the subjectivity detection and polarity detection steps are trained one of these training sets. The other training set's is for language identification. The construction of this dataset and experiment setup for the language identification step is described separately in Section 3.3.1 as we use completely different data and experiments for this step. We do not mention any training set for POS-tagging as the models for POS-tagging are those presented on the website of the Stuttgart TreeTagger and are trained on various corpora. The training set next described is hence only applicable to subjectivity detection and polarity detection.

Our training set needs to adhere to some constraints to suit our needs. We need messages in multiple languages, multiple sentiments and multiple domains. The messages should also stem from social media. Our sources are Twitter, Facebook and Hyves. The methodology we use to collect data covering different sentiments is adopted from [Go et al., 2009] and [Read, 2005], who mention using smilies as noisy labels for sentiment and using new messages as noisy indicators of objective messages. For positive messages we query Twitter for exactly 30 minutes with happy smilies such as *:), :-), :D etc.*. For negative messages we query Twitter for exactly 30 minutes with sad smilies such as *:(, :-(, :'(etc.*. For objective messages, we extract all messages produced by news instances such as the *BBC*, *CNN* (English) or *EenVandaag* (Dutch). We do this again for 30 minutes. Words indicating the nature of the objective message (such as *news* in English, *nieuws* in Dutch) are stripped from the messages. Twitter-specific entities such as usernames, channel names and links are also stripped as this data is also to be applied on different social media. As collecting this training data is a fully automated process, we can very quickly construct a large training set without much effort, something typically useful for a training set. The easy collection of our training data comes at a cost though. As mentioned, the smilies are *noisy* labels of subjective text. The news instances are also *noisy* labels of objectiveness. Smilies are also often used to express irony or sarcasm, thus actually yielding a sentiment opposite of the message's sentiment. News instances are expected to present the news in an objective but often use subjective stances in their articles and even headlines, which are typically what is being said on social media.

Facebook and Hyves both do allow us to query for smilies. Tracking news instances however is much harder for two reasons. The first is the lack of clarity on which instances are news instances. The second is the lack of news instances at all. Due to these limitations, we only use Twitter to extract our training set from. Note that as Twitter messages are typically contain less characters than Hyves and Facebook messages, sentiment expressed in Twitter messages is often more concise than sentiment expressed in

Hyves or Facebook messages. We assume that this implies that the typical way sentiment is expressed on Hyves and Facebook subsumes the typical way sentiment is expressed on Twitter. Hence, we assume that if we train on Twitter sentiment, we can generally capture the core of sentiment expressed on Hyves and Facebook.

For both subjectivity as well as polarity detection we train language specific models. Moreover, the models use part of speech tags as features known beforehand. To this end we use our language identification step to filter out those messages that are neither in Dutch nor in English when querying Twitter with smilies. Note that our language identifier is trained on different data described in Section 3.3.1. This is because we can train our language identifier on more extensive data incorporating more languages since constructing training data for language identification is much easier than it is for subjectivity and polarity detection. All the resulting messages – both subjective and objective – are processed by the POS-tagger to obtain the parts of speech. The size of the resulting training set is shown in Table 9.

As our polarity detection step uses the RBEM algorithm, all patterns for our RBEM models need to be manually labeled from the training data. This process is not trivial and labor-intensive. To this end we do not fully utilize the training set for polarity detection but only use a small fraction thereof. The numbers of patterns present in our RBEM model used for all experiments are shown in Table 8. The amount of patterns shown in Table 8 would be insufficient in a practical setting. For our experiments however, we merely demonstrate the application and strength of the RBEM algorithm by using such little patterns.

Table 8: The number of patterns present in the English and Dutch models of the RBEM algorithm.

English		Dutch	
Type	Count	Type	Count
Amplifiers	67	Amplifiers	29
Attenuators	12	Attenuators	3
Rightflips	38	Rightflips	7
Continuators	10	Continuators	4
Leftflips	5	Leftflips	2
Negatives	532	Negatives	310
Positives	302	Positives	141
Stops	0	Stops	2

3.2.4 Validation Set

Goal 1 requires us to compare different approaches to the same problem with each other. Since little data often implies weak conclusions, we require a large yet accurate dataset to evaluate upon. To this end we use the validation set shown in Figure 6.

To create this validation set, we use a part of the data described in Section 3.2.1 to achieve **Goal 4**. We do this by selecting exactly 500 messages per social medium labeled Dutch by our language identification algorithm. We do the same for English. We manually inspect this label and if it indeed is Dutch or English, we manually determine whether it is a positive, negative, bipolar or objective message. If we are unsure about the true sentiment, the message is left out. As part of speech tags are merely used as features and we do not experiment with the POS-tagger, we do not manually construct these. We rather let our part of speech tagging algorithm determine the tag. This is not a problem as we do not evaluate POS-tagging.

The resulting size of the validation set as used to achieve **Goal 1** is shown in Table 9. Note that the varying sizes of the validation set with respect to social medium and language is due to our certainty of messages actually being what we label them. Facebook for example contains a lot of lengthy messages with a lot of slang in them. This prevents a human being from accurately determining the sentiment of the message.

From our labeling, it turned out that Facebook contained very little Dutch messages. Likewise, Hyves hardly contained any English messages, the English messages that were present were most often cited

from Twitter. Due to these observations, we decided not to include Dutch Facebook messages nor English Hyves messages into our validation set.

Note that we have bipolar messages our validation set. Since we do not have a proper way to obtain such messages in an automated way, this type of messages is not present in our training set and hence our models cannot take this into account. Even though for our problem formulation, we do not care about the bipolar class, we will propose some heuristics to deal with this whenever possible to demonstrate possible ways to incorporate this class.

Table 9: The sizes of the training and validation set. As we cannot determine whether a message is bipolar or not, the bipolar class is missing in the training set. The different sizes for each language and social medium of the validation set is because we only include messages of which we are sure the label is correct.

	Training set		Validation set			
	English	Dutch	English		Dutch	
			Twitter	Facebook	Twitter	Hyves
Positive	3614	1202	94	42	75	71
Negative	3458	1504	77	34	99	82
Bipolar	NA	NA	22	4	15	17
Objective	4706	2099	180	47	111	104
Total	11778	4805	373	127	300	274

3.2.5 Test Set

Achieving both goals **2** as well as **3** requires us to compare one situation with another. More specifically, for **Goal 2** we need to compare our complete process against a more generic process where one step is left out. For **Goal 3** we need to compare a setting where we use our presented solutions for each step against other solutions. As the only thing we want to compare in both situations are exactly these differences, we cannot afford to have incorrect data in our evaluation as this may yield other differences. Consider for example having a message in our dataset wrongfully labeled regarding its language. If we then assume this to be correct and evaluate the outcome of our process on this message, we cannot tell anything about the actual accuracy of the process as we do not even know the real label ourselves. Moreover, introducing an error in the language label may then even correct our dataset rather than introduce errors. To make fair comparisons we thus need a dataset that is completely correctly labeled. To this end, we use the test set shown in Figure 6.

While our validation set may contain some vague cases and possibly incorrect POS-tags, for our test set this is not allowed. To create the test set, we scrape all public Twitter messages in a time period that does not overlap with the period in which the training set's data was collected. Note that as we scrape all public data, the collection of our test set is on a much broader source than our training data is as for our training data we target all publicly available data with smilies in them along with news accounts. We go through these messages to find those written in English or Dutch, aided by our language identification algorithm. We determine whether a message is positive, negative or objective and only include those messages of which we are highly certain belong to a given class. We perform this process until we have exactly 20 positive messages, 20 negative messages and 20 objective messages for both English and Dutch. The resulting test set thus contains 120 messages in total.

The resulting test set is duplicated. In this duplicate version we correct all grammatical and spelling errors without altering the number of words and their order. Hence not all grammatical errors can be corrected as some errors require us to split words, which is not allowed. An example is *idontknow*, which should be correctly phrased *I don't know*. On this corrected duplicate corpus we run the POS-tagger used in our sentiment analysis process. The resulting tags are more accurate as the algorithm is not troubled by grammatical errors. We manually go through all POS-tags and correct any wrongfully assigned tags. The resulting POS-tags are then assigned to our original, grammatically uncorrected dataset. This way we maintain our original data but do obtain POS-tags more accurately.

We note that the relatively small size of the test set may jeopardize our results and their validity but this

is a resource constraint as labeling is a very labor-intensive process[9]. The same holds for the contents of the test set. As we need it to be as accurate as possible, any (extremely) vague cases which are close to the decision boundary are not included due to uncertainty.

3.3 Comparative Experiments

We compare and analyze our sentiment analysis approach on the level of each single step and on the level of the complete process, comprising all steps. On the level of a single step, we compare each of the algorithms used in the separate steps against other algorithms being either competitor approaches or baslines. This way we achieve **Goal 1**. When looking at the global level, we investigate the addition each step brings to solving the complete sentiment analysis problem. This achieves **Goal 2**. In addition we also quantify the effect of having each step as accurate as possible to achieve **Goal 3**.

All metric scores presented in this section, with exclusion of the language identification experiments in Section 3.3.1, are the mean of four scores. We present only these mean scores for conciseness but we do mention any peculiarities in different social media or languages when applicable. These four scores stem from combinations of social media and languages. As Hyves is a Dutch social network, it mainly contains Dutch messages. This is our first combination. During the construction of the datasets presented in Section 3.2, we found that Facebook contains very little (publicly available) Dutch messages and hence we only regard English messages from Facebook, being our second combination. The third and fourth combinations consist of Dutch Twitter messages and English Twitter messages. For each experiment we perform it on each of these four combinations and report the mean of the presented metric thereof.

We report three different metrics; *accuracy*, *precision* and *recall*. We also make use of the *F-measurement* being the harmonic mean between precision and recall. The F-measure is not reported here, as it can be deduced from the precision and recall, but only used to directly compare two results. If we define the set of possible classes as C and our dataset as the set of instances I then these metrics are computed as the mean among all classes in C as follows.

$$Accuracy = \frac{\Sigma_{c \in C} |correctly\ classified\ as\ c|}{|I|} \tag{3}$$

$$Precision = \frac{\Sigma_{c \in C} \frac{|correctly\ classified\ as\ c|}{|all\ documents\ classified\ as\ class\ c|}}{|C|} \tag{4}$$

$$Recall = \frac{\Sigma_{c \in C} \frac{|correctly\ classified\ as\ c|}{|all\ documents\ in\ c|}}{|C|} \tag{5}$$

$$Fmeasure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{6}$$

Note that these metrics are all aggregated over the four different combinations of language and source of social medium that we have in our dataset for compactness. We consider an example calculation of each of the metrics to illustrate the calculation. Consider having the confusion matrix shown in Table 10. The table contains the classification results on all four combinations of social media and languages we regard. We now compute the accuracy, precision, recall and F-measure as follows.

$$\begin{aligned} Accuracy &= \frac{\Sigma_{c \in C} |correctly\ classified\ as\ c|}{|I|} \\ &= \frac{110 + 80 + 20}{110 + 20 + 10 + 5 + 80 + 20 + 40 + 50 + 50} \\ &= 0.545 \end{aligned}$$

[9]Manually labeling the 120 messages in our test set took us two full working days. An expert linguist may do this in half the time though.

$$
\begin{aligned}
Precision = & \frac{\Sigma_{c \in C} \frac{|correctly\ \ classified\ \ as\ \ c|}{|all\ \ documents\ \ classified\ \ as\ \ class\ \ c|}}{|C|} \\
= & (\frac{110}{110+5+40} + \{for\ the\ positive\ class\} \\
& \frac{80}{20+80+50} + \{for\ the\ negative\ class\} \\
& \frac{50}{10+20+50})/3 + \{for\ the\ objective\ class\} \\
= & 0.639
\end{aligned}
$$

$$
\begin{aligned}
Recall = & \frac{\Sigma_{c \in C} \frac{|correctly\ \ classified\ \ as\ \ c|}{|all\ \ documents\ \ in\ \ c|}}{|C|} \\
= & (\frac{110}{110+20+10} + \{for\ the\ positive\ class\} \\
& \frac{80}{5+80+20} + \{for\ the\ negative\ class\} \\
& \frac{50}{40+50+50})/3 + \{for\ the\ objective\ class\} \\
= & 0.635
\end{aligned}
$$

$$
\begin{aligned}
Fmeasure = & \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \\
= & \frac{2 \cdot 0.639 \cdot 0.635}{0.639 + 0.635} \\
= & 0.637
\end{aligned}
$$

Table 10: An example confusion matrix. The labels are the real labels of an instance whereas the classes are what the classifier classified an instance to be.

	Label Positive	Label Negative	Label Objective
Class Positive	110	5	40
Class Negative	20	80	50
Class Objective	10	20	50

3.3.1 Single Step Comparison

To achieve **Goal 1**, we compare each of the algorithms we use for each step against known baselines and other approaches proposed in literature. We do not do this for POS-tagging as experiments on POS-tagging are presented in [Schmid, 1994]. For language identification, we use different datasets. We do this because we can more easily construct labeled data for language identification, making our results stronger, than we can for subjectivity and polarity detection. For subjectivity and polarity detection, each baseline or approach is trained on the training set described in Section 3.2.3 unless otherwise mentioned. The validation set for the subjectivity and polarity detection is always the validation set described in Section 3.2.4. If the language of a message or its POS-tags are required in advance (for subjectivity or polarity detection), we use the language identification algorithm and POS-tagger used in our sentiment analysis process to determine these features.

The approaches we compare against are the following. Note that to keep the presentation compact, we intentionally do not describe well known state-of-the-art approaches such as Nave Bayes and SVM in detail. However, we do provide essential details for the clarification and reproducibility of the experimental study.

Step 1. Language Identification

Approach 1 - N-gram based approach of [Cavnar and Trenkle, 1994] using the cumulative frequency based measurement of [Ahmed et al., 2004]. This is a direct competitor of LIGA and is the current state-of-the-art approach.

Step 2. Subjectivity Detection

Approach 1 - Majority class guess. This approach simply assigns the class prevailingly occurring in the training set to any instance in the validation set. The model this method learns is hence a simple function $f : T \rightarrow c_m$ where T is the set of all texts and c_m is the majority class. It only makes sense to use such a classifier when we have an imbalance in class representation in our training set, which is the case. This can be seen from the differing sizes of the objective and subjective classes in Table 9.

Approach 2 - Naive Bayes classifier. Using four different variants of features: tokens, tags, a combination of the two and patterns. Naive Bayes is a competitor approach as it often shows promising results [Pak and Paroubek, 2010, Go et al., 2009] in any text categorization problem.

Approach 3 - Support Vector Machines. Similar to Naive Bayes, SVM approaches often show very promising results in traditional text categorization [Joachims, 1998], and more specifically in subjectivity detection [Pang et al., 2002, Esuli and Sebastiani, 2006], this approach is hence a direct competitor. In our experiments we use the same parameter settings as [Pang et al., 2002, Esuli and Sebastiani, 2006], who used the default settings. We use the same feature spaces as for Naive Bayes, using tokens, tags, a combination of both or patterns as features.

Approach 4 - RBEM. This is the Rule-Based Emission Model used for polarity detection. We investigate its applicability to subjectivity detection. As polarity detection and subjectivity detection are often combined, this is also a competitor approach.

Approach 5 - Prior polarity classifier. This naive classifier is one of the simplest and straightforward approaches. It takes as input two lexicons containing positive and negative words. These are the prior (out-of-context) polarities of these words. This classifier solely uses these word lists and matches texts against them. If no match can be made, the text is assumed to be objective, otherwise it is subjective. This classifier is a baseline as, in contrast to for example the majority class guess, it requires no training on labeled data but merely a lexicon.

- As a special case of a prior polarity classifier, we will complement the default prior polarity classifier with SentiWordNet 3.0[10] [Esuli et al., 2010] as a lexicon in an attempt to find more polarized words.

Step 3. Polarity Detection

Approach 1 - Majority class guess. Always assigns the label of the majority class of the training set. The motivation for this baseline again comes from the imbalance in positive versus negative data.

Approach 2 - Naive Bayes classifier. Using four different variants of features: tokens, tags, a combination of the two and patterns. In line with subjectivity detection, naive Bayes approaches often do well in polarity detection. This is hence a competitor approach.

Approach 3 - Support Vector Machines. Often shows very promising results and is hence a competitor. We use the same settings as for Naive Bayes.

Approach 4 - AdaBoost classifier. AdaBoost is used as selected approach for subjectivity detection but we also investigate its applicability to polarity detection partly because we have this approach at hand. With similar reasoning as for applying the RBEM algorithm to subjectivity detection, that subjectivity and polarity detection are often combined, this is a competitor rather than a baseline.

Approach 5 - Prior polarity classifier. This classifier is the same as the one described in Step 3, Approach 5

Language Identification

As we use different data and experiment setups for the language identification step, we first describe the dataset, next outline the experiment setup and finally present the results. To disambiguate the training

[10] http://sentiwordnet.isti.cnr.it/ - A lexical resource for opinion mining.

and test set used for language identification from those used in subjectivity and polarity detection (as shown in Figure 6 and Table 4) we refer to the training and test set for language identification as the training$_{LI}$ and test$_{LI}$ set respectively.

The experiments on language identification include six languages as this will give us stronger results than just using Dutch and English, which is what we do for subjectivity and polarity detection. We can do this as obtaining labeled data for language identification is much easier than it is for subjectivity and polarity detection. The models learned on 50% in these language identification experiments are the ones we use in the succeeding experiments, filtering out any language other than Dutch or English by using the threshold presented in Section 2.1.3.

Similar to our training set for subjectivity and polarity detection, we only use data from Twitter to train upon. For subjectivity and polarity detection we do evaluate on data from other social media but for language identification we do not do this. We assume that characteristics of language on social media do not differ per social medium. Note that this on social media, we do not state that for example English language used on a social medium contains the same characteristics as English used in news articles.

Dataset and Experiment Setup
The dataset was constructed from the social medium *Twitter*. The Twitter API is used to extract messages from accounts known to only contain messages of a specific language. We do this for six languages and six accounts per language. The six languages are German, English, Spanish, French, Italian and Dutch. These are languages we have sufficient knowledge of to identify. Moreover, including Spanish, French and Italian presents a challenge as these languages contain a lot of similar word extensions and trigram patterns. For every language we have at least one account of a person instead of an institution (such as BBC News). We assume that each account has its own domain and hence its own jargon that typically is not, or less often, used in the other domains. When we indicate a domain as *random* we mean that its messages are a mixture of other domains. The rationale behind using six accounts per language is that our experiments require us to have different domains but incorporating tens or hundreds of accounts is very laborious.

As a pre-processing step, we inspect our data as in [Cavnar and Trenkle, 1994], removing messages that contain multiple languages or bilingual terminology. From each message we also deliberately remove links, usernames preceded by an @ sign, term references preceded by a # sign or smilies such as *:)* and punctuation. The rationale behind this is that we want to learn a model on the language itself whereas these entities are language-independent. Moreover, the use of Twitter for our experiments is just to show an application on short texts, learning Twitter-specific patterns containing username or channel references is not desired. The final dataset contains 9066 labeled messages of at most 140 bytes.

Table 11: Splitting the data into training$_{LI}$ and test$_{LI}$ sets.

Lang.	Domain	Exp. 1		Exp. 2
DE	Sports	s_1 ... s_n	$train_1$ / $test_1$	$test2_{gen}$
	National news	n_1 ... n_n	$train_1$ / $test_1$	$train_2$ / $test2_{spec}$
	Random	r_1 ... r_n	$train_1$ / $test_1$	$test2_{gen}$
NL	Telecom.			
...	...	...		...

In both approaches we use trigrams (as suggested in [Cavnar and Trenkle, 1994]) and the frequency based measurement of [Ahmed et al., 2004].

In all cases we compare the mean accuracy of the N-gram approach against the mean accuracy of LIGA. The mean accuracies are obtained by taking the mean of 50 different ten-fold cross-validation experiment runs. We check for statistical significance using pairwise T-tests.

In order to compare the accuracies of both methods, we learn a model on one part of the data, which we call $train_1$, and test it on another part of the data, called $test_1$ formed as illustrated in Table 11, column Exp. 1. We use all accounts of all languages. The size of $train_1$ varies to investigate the influence of the corpus' size. We use 5%, 10%, 25% and 50% of the entire dataset stratified per language and sampled uniformly.

We also investigate how much each approach learns about the actual language itself rather than about a particular domain. To analyze this, we learn a model on data from one domain and test it on other domains. For each language, we choose a single account on which we learn our model and then test on the remaining accounts (Table 11, column Exp. 2). The training$_{LI}$ set $train_2$ consists of $\frac{2}{3}$ of all data of one single account for each language. There are now two test$_{LI}$ sets. The first, $test_{2_{spec}}$ is the remaining $\frac{1}{3}$ of the same account and will allow us to judge how specific each model is for a single domain. The second, $test_{2_{gen}}$ consists of all other accounts. This test$_{LI}$ set will show us how well the learnt model generalizes to other domains.

Finally, we analyze the influence of jargon. To study this, we learn a model on all accounts of all languages except for one or two hold-out accounts which are reserved for testing. Assuming that each account represents a single domain, we can get to know how important it is to capture domain-specific jargon since any domain-specific jargon present in one of the holdout accounts likely will not be included in our model. The formation of the training$_{LI}$ and test$_{LI}$ datasets can be seen as the inverse of Table 11, column Exp. 2.

Experiment Results
The main results are presented in Table 12 which shows, averaged over 50 experiment runs, the accuracies and standard deviations for LIGA and N-gram approaches. The last column shows the result of pairwised T-test (which was positive for each comparison in our case).

Table 12: Accuracies averaged over 50 runs of ten-fold cross-validation.

Experiment	LIGA	N-gram	t-Test
5% sample	94.9 ± 0.8	87.5 ± 1.5	√
10% sample	96.4 ± 0.5	90.6 ± 1.0	√
25% sample	97.3 ± 0.5	92.5 ± 0.9	√
50% sample	97.5 ± 0.5	93.1 ± 0.8	√
5% vs. 50%	94.9 ± 0.8	93.1 ± 0.8	√
Generalization	92.4 ± 1.0	83.5 ± 1.8	√
Specialization	98.3 ± 0.8	95.5 ± 1.6	√
One holdout	95.6 ± 1.6	89.2 ± 4.3	√
Two holdouts	95.2 ± 0.9	88.1 ± 2.7	√

Effect of the training$_{LI}$ set size. As expected we can see that as the size of the training$_{LI}$ data grows, the accuracy increases and variance decreases with both approaches. However, LIGA has much better performance than N-gram approach especially when a small amount of labeled data is available.

The row '5% vs. 50%' of Table 12 compares LIGA learned on 5% of the training$_{LI}$ data against the N-gram-based approach using 50% of the training$_{LI}$ data. LIGA still has statistically significant higher accuracy.

It can be clearly seen from Figure 8 that LIGA *consistently* outperforms N-gram approach on each of the 50 experiment runs. To avoid overloading of the graph we present only accuracies corresponding to the use of 5% and 50% of available labeled data.

Using more than 50% of labeled data for training$_{LI}$ a language identification model in our case did not result in any further significant improvement of the classification accuracy for either of the two approaches and therefore we omit these results for the sake of conciseness.

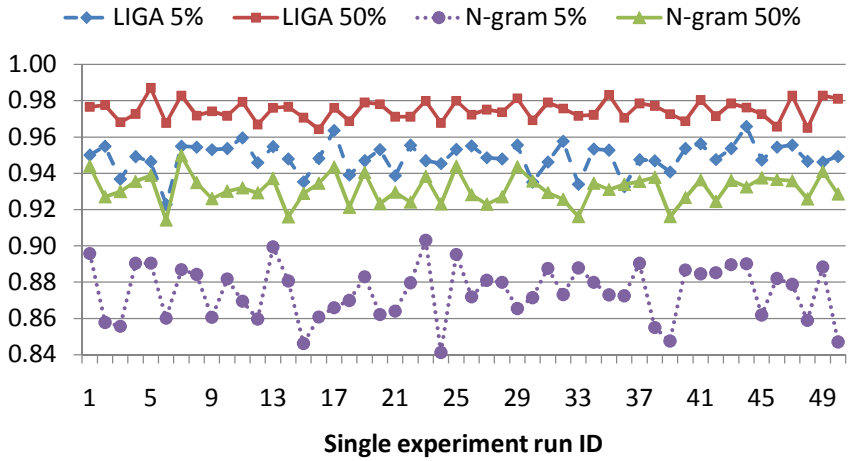


Figure 8: Accuracy results over 50 different runs for LIGA and N-gram approaches when trained either on 5% or on 50% of available labeled data.

Generalization and domain-specific jargon results. The 'Generalization' and 'Specialization' rows in Table 12 and Figure 9 show the results from the generalization experiments. As one would expect, a model learned on a specific domain yields higher accuracies in that domain than in other domains. This behavior is shown by the difference in accuracies between the generalization results and the specialization results. The specialization accuracy for both approaches is often close to 100%, yet LIGA is statistically significantly better than N-gram approach.

For the generalization experiments, we clearly see that LIGA again consistently (Figure 9) outperforms the N-gram based approach. The accuracy of LIGA never drops below 90% whereas that of the N-gram based approach is never higher than 90%.

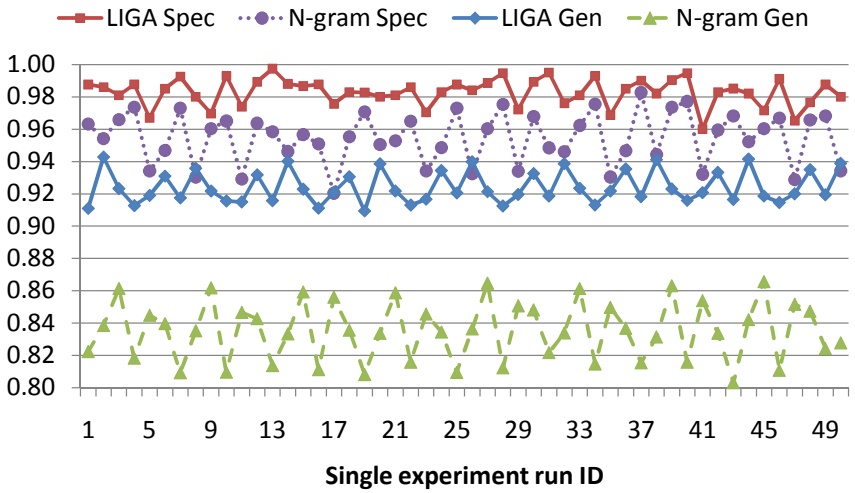


Figure 9: Accuracy results over 50 different runs for LIGA and N-gram approaches when trained on one domain and tested on all other domains (see Table 11, column Exp. 2 for clarification).

LIGA achieves higher accuracies both within the domain the model was trained on as well as outside that domain. This indicates that LIGA is likely to be less sensitive to domain-bound usage of language and hence is likely to learn more about the language itself rather than about the domain messages belong to.

The results plotted in Figure 10 stem from holding out one or two accounts. Averaged results are in the last two rows of Table 12. We can observe a moderate drop in accuracy for both approaches with respect to not holding out any account. LIGA's accuracy drops by 2-3% on a single holdout and the

N-gram-based approach's accuracy drops by 4-5%. Testing on two holdout accounts decreases accuracy a bit further but not much. This indicates that the use of domain-specific jargon introduces an error of around 3% for LIGA and 5% for the N-gram-based approach. LIGA consistently outperforms the N-gram-based approach.

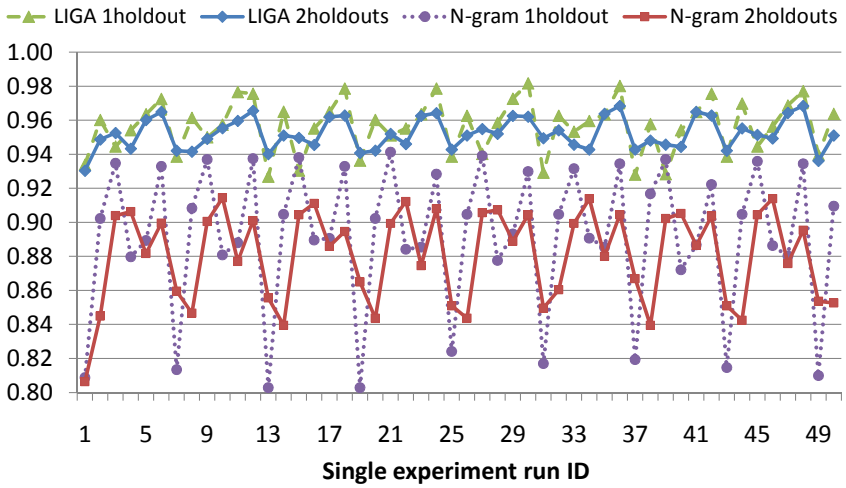


Figure 10: Accuracy results over 50 different runs for LIGA and N-gram approaches with one and two hold out accounts.

We observe that for some holdout accounts the accuracy is always lower than for others. There are thus some domains that use more jargon than others. However, across multiple experiment runs, the variance in accuracy is always much larger for the N-gram approach.

Subjectivity Detection

We perform subjectivity detection using AdaBoost with decision stumps as weak learners. We compare using AdaBoost against four other methods tackling subjectivity detection. We describe each method briefly and discuss their results which are presented in Table 14. Note that in the midsection of Table 14 we show accuracies on all data (the row titled *with missed*) as well as accuracies where we leave out all texts that are subjective but could not be identified as such due to insufficient labeling (the row titled *without missed*).

Majority Class Guess – From Table 9 we see that the English training set comprises 7072 subjective messages and 4706 objective messages. The majority class is hence the subjective class. For Dutch we have 2706 subjective messages and 2099 objective messages. Again the majority class is the subjective class. Using these two majority classes, we achieve a mean accuracy of 58.9% having a precision of 29.4% and a recall of 50%.

Naive Bayes – We train a Naive Bayes classifier using four types of features. The first type contains word tokens, yielding a bag of words. The second type contains tags, yielding a bag of tags. The third type combines both tokens and tags by taking the union of the previous two feature sets. The last feature set we look at consists of patterns. Patterns consist of any possible combination of tokens and tags. The only restriction is that the ordering in the pattern must adhere to the ordering of the original message. For our experiments we will use unigrams, bigrams, trigrams and quadrigrams. Since we use binary occurrence flags instead of frequencies or TF-IDF scores[11] for our AdaBoost approach, we also only regard using binary occurrence flags for Naive Bayes.

As an example showing pattern features we regard the sentence of token/POS-tag pairs, *The/DT nice/JJ car/NN*. For tokens we transform all characters to lower case. We then get the bigram feature spaces shown in Table 13. The uni-, tri- and quadrigram feature spaces are similar but then contain all possibilities for a one-, three- and four-sized window instead of a two-sized window.

[11] Term frequencyinverse document frequency, a frequency metric often used in natural language processing evaluating how important a term is to a document with respect to the entire corpus.

Table 13: The bigram feature spaces for using tokens, tags, a mixture of both and patterns.

Feature Space	Example
Tokens	(the,nice),(nice,car)
Tags	(DT,JJ),(JJ,NN)
Mixture	(the,nice),(nice,car),(DT,JJ),(JJ,NN)
Patterns	(the,nice),(nice,car),(DT,JJ),(JJ,NN), (DT,nice),(JJ,car),(the,JJ),(nice,NN)

Since the feature space can become very large, we investigate using those features that yield the most information. To do this, we use mutual information and select the x most informative features as a selection criterion. The mutual information for a feature f is computed as follows.

$$I(U,C) = \sum_{f\in\{0,1\}} \sum_{c\in\{0,1\}} P(U=f, C=c) log_2 \frac{P(U=f,C=c)}{P(U=f)P(C=c)}$$

Mutual information. U is 1 if f occurs and 0 otherwise and C is 0 or 1 depending on whether the text is subjective or objective respectively.

In Table 14 the top section shows the results of the Naive Bayes algorithms using different feature sets. Results are listed for the case when all features or the top 2000, 4000 or 8000 features are used. Note that we do not attempt to optimize the number of features used but rather investigate different settings to gain insights into the effect of using different features.

In all cases, using POS-tags as features is favored by our experiments, yielding accuracies up to 7.1% higher than the majority class guess. When using 8000 features, we achieve higher accuracies than using 2000, 4000 or all features. This indicates that more features yield more information but at some point there is a turn where too much noise (meaningless features) is included.

The precision of all Naive Bayes approaches is not really promising but much higher than that of the majority class guess. The recall however is drastically low, indicating that we are often not able to return all messages we are interested in. As we regard recall of all classes, we have a lot of subjective messages being classified as objective and vice versa.

Support Vector Machines – Support vector machines (SVM) map the N-dimensional feature space used onto an (N+1)-dimensional feature space in an attempt to linearly separate the instances into two classes in this higher dimensional space. For our experiments we use the SVM^{light} implementation of [Joachims, 1999][12] with just the default settings which is also typically done in literature [Pang et al., 2002, Esuli and Sebastiani, 2006].

The different feature spaces and the number of features used are equal to those used for the Naive Bayes classifier. We experiment with using tokens, tags, a mixture of the two and patterns as features and use either all features or only the top 2000, 4000 or 8000 of them according to the mutual information criterion.

The results of all SVM experiments for subjectivity detection are listed in the upper midsection of Table 14. Similar to what we see with the Naive Bayes approaches, using all features results in better metrics for the SVM approaches. Generally speaking, the more features are used, the higher the precision values become but the lower the recall becomes. There is hence a clear tradeoff between precision and recall where using all features instead of a subset of them results in the best balance for the settings we experimented with.

The experiments favor using patterns and all features, yielding an increase in accuracy of 15.2% over the majority class guess and 11% over the Naive Bayes approach using 8000 features and a mixture of tokens and tags. The precision and recall however are much higher than they are for the majority class guess and the Naive Bayes approaches.

[12] The SVM^{light} implementation can be downloaded at http://svmlight.joachims.org/

RBEM – RBEM is the algorithm we use for polarity detection. We apply this algorithm to the subjectivity detection problem as we have it at hand, out of curiosity how it performs and because subjectivity and polarity detection are often combined. Since we only care about presence of subjectivity and not about its polarity, we let the negative patterns emit positive sentiment. We then determine a positive threshold and classify a message as subjective if this threshold is exceeded. We classify the message as objective otherwise. By letting negative patterns emit positive sentiment we prevent highly subjective messages in which negative patterns rule out positive patterns from being classified as objective.

We experimented with different thresholds ranging from 0 to 2.0 with steps of 0.1. It turned out however that using 0 (i.e. any sentence with any emission value is subjective) yields the best results for all metrics, among all tested thresholds and hence only these results are shown in Table 14.

The RBEM algorithm requires us to extensively label training data. Since this is a very labor-intensive process, we only label a fraction of the entire training set. We report the accuracies for the case where we incorporate texts that we know are subjective but of which none of its subjective patterns are present in our model as well as the case where we leave these out as we expect that more extensive labeling will result in better performance. This is investigated when we evaluate the single step experiments for polarity detection.

The results of applying RBEM to subjectivity detection are shown on the left of the lower midsection of Table 14. The accuracies are higher than those of the majority class guess as well as all Naive Bayes approaches, yielding improvements of 13% and 8.% respectively. The results of the SVM approach however are not matched on either of the three metrics we regard. Facebook and Hyves texts are generally longer than Twitter texts which explains the higher values of missed out patterns for these two social media.

More important than the accuracies however are the precision and recall metrics for the RBEM algorithm. Where the majority class guess scored very low on precision and the Naive Bayes approach very low on recall, the RBEM algorithm performs very well but still not better than the SVM approach.

Prior Polarity Classifier – A prior polarity classifier only looks at the prior polarity of words; the polar orientation of a word on its own, without any context. We use three lexicons, a positive and negative one, containing words with their prior polarity and a list with negation words. A negation word flips the polarity of a prior polarity word when it directly precedes the prior polarity word. When the number of positive and negative terms exceed a certain threshold we say that the message is subjective, it is objective otherwise. The lexicons we use result from the manual labeling for the RBEM algorithm used for polarity detection. Since we have combinations of tokens and tags present in our lexicon, we will compare against using just tokens as well as using tokens in combination with tags.

As the lexicon used in the RBEM algorithm is relatively small in size, we also investigate using SentiWordNet 3.0 as a lexicon of tokens only. We use a word's prevailing polarity in SentiWordNet to determine its polarity. We sum the absolute values of the scores of all words present in the message and when this exceeds a certain threshold, we classify the message as subjective. We classify it objective otherwise.

For the prior polarity classifiers not using SentiWordNet we count the fraction of entities found to be polar. Whenever this fraction exceeds a threshold, we classify the message as being subjective and objective otherwise. We do not attempt to find an optimal threshold but to put some effort into finding a relatively good one. We do so by first investigating thresholds ranging from 0 to 1.0 with steps of 0.01. We then find the two best performing thresholds t_1 and t_2 as determined by the F-measurement and further examine thresholds in between t_1 and t_2 but now with steps of 0.001. We then report the threshold that shows the best F-measurement. For the token-based prior polarity classifier the threshold is 0.001. For the token and tag-based classifier the threshold is also 0.001. For the SentiWordNet classifier the threshold is 0.005.

The right half of the midsection of Table 14 shows the accuracies for a prior polarity classifier with differing lexicons. From Table 14 we see accuracies that seem promising at first sight for the regular prior polarity classifiers. However, when we look at the amount of messages missed out, we see relatively high values. Using SentiWordNet (SWN) expands our lexicon and hence less messages are missed out

but the accuracy drops drastically as well, resulting in an even worse accuracy than the majority class guess. Note that the downloadable version of SentiWordNet 3.0 only contains tokens and no POS-tags. It is very likely that using SentiWordNet with POS-tags increases accuracy. Moreover, SentiWordNet is not available in Dutch and hence the accuracy shown in Table 14 is the mean of the accuracies on the English Twitter and Facebook validation sets.

The precision and recall values for the non-SentiWordNet prior polarity classifiers are relatively steady. When using SentiWordNet however the recall is relatively low when compared to its accuracy. This strangely enough indicates that we miss out relatively more required messages than with the non-SentiWordNet prior polarity cases, even though our lexicon is expanded. This exact reason why this happen is not clear but it could be due to the relatively overall performance of the SentiWordNet classifier.

AdaBoost – AdaBoost is the algorithm we use for subjectivity detection. The features we use are the unions of uni-, bi-, tri- and quadrigrams of POS-tags but for completeness we also report the results when using only tokens. Note that in case of POS-tags, we do not use the prior polarity lexicons as described in Section 2.3.3.

AdaBoost uses x weak classifiers to create a strong classifier. This x is its only parameter. We experiment with varying values for x as shown in the bottom section of Table 14. Using as little as 25 weak learners already yields accuracies of almost 75%. Using 50 weak learners yields a mean accuracy of 76.2%, an improvement of 17.3% over the majority class guess. Using more than 50 learners decreases accuracy, indicating that we are overtraining our models.

The precision and recall of AdaBoost are much better than those of the majority class guess and the Naive Bayes approaches and slightly better than those of the SVM and RBEM approaches. The precision is a bit higher than the recall however, indicating that we can rather say that what we return is correct instead of that what is correct is also returned.

Polarity Detection

To perform polarity detection, we use the RBEM algorithm. For completeness, we again briefly describe each approach we compare against. The results of these experiments are shown in Table 17, again also listing all metrics in the midsection where we either take missed out messages into account or not.

Since we want to analyze only the polarity detection qualities of all approaches, we leave out all objective texts from the training and validation sets because these texts have no polarity. Our training set contains only positive and negative messages whereas our validation set additionally contains bipolar messages. We can thus not straightforwardly classify a message to be bipolar as there is no such data present in our training set. We will use heuristics to overcome this issue whenever applicable.

Majority Class Guess – This classifier assigns the prevailingly occurring class of our training set to each unlabeled instance it needs to classify. The difference in using a majority class guess for polarity detection with respect to subjectivity detection are the present classes. We now have a positive and negative class. As there is no straightforward way to include the bipolar class we always misclassify these type of messages. The majority class for English is positive whereas for Dutch it is negative. From Table 9 we can deduce a mean accuracy of 50.2%. The mean precision and recall are 14.9% and 33.3% over all classes and social media respectively, much lower than the accuracy. Note that for English the recall is 100% for the positive class whereas for Dutch the recall is 100% for the negative class. The precision and recall for all other classes are always 0% however, explaining the much lower precision and recall with respect to the accuracy.

Naive Bayes – Applying a Naive Bayes classifier for polarity detection is similar to applying it for subjectivity detection. A Naive Bayes classifier assigns a probability to each class when classifying a previously unseen message We again experiment with using just tokens, using just tags, using both tokens and tags and using patterns as feature sets (Table 13).

To cope with the bipolar class, we set a threshold for both the positive as well as the negative class. We then say that whenever both probabilities exceed this threshold, we label the message as being bipolar.

Table 14: The overall accuracy (A), precision (P) and recall (R) of algorithms for subjectivity detection. The instances per algorithm having the highest F-measure are shown in bold. The 'Missed %' row lists the fraction of subjective texts not found subjective due to insufficient labeling.

Naive Bayes	All		MI 2000		MI 4000		MI 8000	
	A	0.650	A	0.613	A	0.616	A	0.608
Tokens	P	0.571	P	0.521	P	0.527	P	0.525
	R	0.399	R	0.331	R	0.347	R	0.358
	A	0.662	A	0.644	A	0.668	A	0.670
Tags	P	0.574	P	0.561	P	0.578	P	0.579
	R	0.350	R	0.348	R	0.353	R	0.358
	A	0.527	A	0.618	A	0.629	**A**	**0.631**
Mixture	P	0.541	P	0.538	P	0.561	**P**	**0.567**
	R	0.602	R	0.364	R	0.407	**R**	**0.427**
	A	0.423	A	0.603	A	0.604	A	0.606
Patterns	P	0.494	P	0.518	P	0.519	P	0.523
	R	0.299	R	0.374	R	0.371	R	0.383

SVM	All		MI 2000		MI 4000		MI 8000	
	A	0.716	A	0.614	A	0.619	A	0.610
Tokens	P	0.736	P	0.610	P	0.615	P	0.612
	R	0.702	R	0.986	R	0.969	R	0.943
	A	0.729	A	0.635	A	0.666	A	0.684
Tags	P	0.734	P	0.636	P	0.695	P	0.714
	R	0.748	R	0.907	R	0.810	R	0.804
	A	0.734	A	0.610	A	0.563	A	0.581
Mixture	P	0.736	P	0.613	P	0.666	P	0.671
	R	0.752	R	0.939	R	0.624	R	0.649
	A	**0.741**	A	0.599	A	0.600	A	0.599
Patterns	**P**	**0.741**	P	0.601	P	0.601	P	0.601
	R	**0.766**	R	0.994	R	0.993	R	0.986

	RBEM		Prior, Tok		Prior, Tok+Tags		Prior, SWN	
	A	0.719	A	0.793	A	0.768	A	0.579
With missed	P	0.720	P	0.742	P	0.763	P	0.549
	R	0.747	R	0.725	R	0.767	R	0.461
	A	**0.733**	A	0.914	**A**	**0.932**	A	0.583
Without missed	**P**	**0.737**	P	0.916	**P**	**0.919**	P	0.556
	R	**0.751**	R	0.992	**R**	**0.993**	R	0.475
Missed %	8.7%		13.1%		17.4%		0.6%	

AdaBoost	25 models		50 models		75 models		100 models		250 models	
	A	0.697	A	0.705	A	0.715	A	0.727	A	0.731
Tokens	P	0.701	P	0.713	P	0.719	P	0.733	P	0.736
	R	0.644	R	0.658	R	0.664	R	0.671	R	0.674
	A	0.743	**A**	**0.762**	A	0.742	A	0.742	A	0.737
Tags	P	0.784	**P**	**0.791**	P	0.788	P	0.788	P	0.779
	R	0.751	**R**	**0.779**	R	0.750	R	0.750	R	0.743

In our experiments however, it turned out that using any threshold almost always resulted in classifying all messages as bipolar. This is because likelihood estimates for both the positive as well as the negative class are very small one for some messages but then very big for other messages. We hence use 0 as a threshold indicating that we never classify any message as bipolar. These type of messages are hence always misclassified.

The top section of Table 17 shows the mean metrics for a Naive Bayes classifier using all features and using the 2000, 4000 and 8000 most informative features determined on mutual information. In contrast to subjectivity detection, polarity detection is best solved using tokens as features. The problem of overtraining we have with subjectivity is leveraged for polarity detection.

The results however are not so promising but still slightly better than the majority class guess. The precision however is much better. Using Naive Bayes with all tokens hence hardly improves accuracy and recall over the majority class guess but does increase the precision by 21.7%, though still not being very high. Similar to what we see with subjectivity detection, we see that the recall is much lower than the precision, indicating that what is classified is correct but a lot of instances are not found.

Support Vector Machines – We also investigate the application of SVMs on polarity detection. Our experiments show that SVMs work well for subjectivity detection and since related work often combines subjectivity and polarity detection into one, it could be that SVMs also perform well on polarity detection.

The bipolar class is included as follows. SVM^{light} outputs a number for each instance for which it holds that if the number is bigger than 0, the message belongs to class 1 (positive in our case) and if it is lower than 0 it belongs to class -1 (negative). We say that whenever the absolute value of this number is below some threshold, the message belongs to the bipolar class. We experimented with different threshold from 0. to 2.0 with steps of 0.1. We next take the two succeeding thresholds t_1 and t_2 that showed best results and experimented with thresholds ranging from t_1 to t_2 with steps of 0.01. The found that using 0 as threshold performed best. Using 0 as threshold means we never classify something as being bipolar, indicating that we cannot properly determine this class but only capture those cases in which is clear what the prevailing sentiment is.

We again experiment with using four different feature spaces where we use either tokens, tags, a combination of both or patterns. We also experiment with using all features or the top 2000, 4000 or 8000 features as ranked by mutual information. The upper midsection of Table 17 shows the metrics of the SVM experiments. The results are not as promising as they were for the subjectivity detection case but better than using a Naive Bayes classifier or a majority class guess, showing improvements of 13.% and 2.1% over the accuracy of the majority class guess and the Naive Bayes classifier.

Even though the accuracy of the SVM approach is close to that of the Naive Bayes approach, the SVM has much higher recall whereas the precisions are also comparable. An SVM approach using all features and patterns performs best among the experimented environments, which was also the case for subjectivity detection. This SVM approach is hence preferred over a Naive Bayes classifier.

AdaBoost – Besides using AdaBoost for subjectivity detection, we also investigate its performance on polarity detection. The scores in the bottom section of Table 17 show the results using varying amounts of weak learners.

AdaBoost's output is a real number of which the sign indicates the class. Similar to what we do for Naive Bayes and SVM approaches, we find a threshold determining that a message belongs to the bipolar class. We use the same procedure as before to determine this threshold and find a value of 0.18.

From Table 17 it follows that using POS-tags and 50 weak learners yields an accuracy of 72.3%, an increase of 22.1% with respect to the majority class guess and the best accuracy among the settings and features we experimented with. The parameter settings are in line with what we see for subjectivity detection.

Even more important than the accuracy are the values for precision and recall, showing much higher values than with the majority class guess, the Bayesian approaches and the SVM approaches and not

differing as extremely from the accuracy as is the case with the Bayesian and SVM approaches.

Prior Polarity Classifier – Using a prior polarity classifier for polarity detection is straightforward, we simply assign the class of which the prior polarity of all words prevails. To cover the bipolar class we use the following heuristic. Whenever the sum of the positive scores as well as the sum of the negative scores both exceed a certain threshold, we classify the message to be bipolar. Determining what thresholds to use is done in the same manner as we do for the subjectivity detection case. We start by experimenting with thresholds between 0 and 1.0, taking steps of 0.01. We then further zoom in on those thresholds that performed best. While for subjectivity we repeat this process only twice, for polarity detection we did this up to 5 times as this kept on showing great differences in performance with each iteration. The resulting thresholds for using just tokens, using both tokens and tags and using SentiWordNet are 1.0^{-6}, 1.0^{-6} and 0.03. Table 17 only lists the results using these thresholds. We again stress that we do not attempt to find an optimal threshold but just compare a number of them to find a reasonably representative good performing solution.

The results for the prior polarity classifiers are shown in the right half of the midsection of Table 17. The number of messages missed out is the same as it is with subjectivity detection but since we disregard objective messages, the fraction of such messages increases. The accuracies are more promising than for subjectivity detection, 13.5% better than the majority class guess when using SentiWordNet, but not better than the AdaBoost algorithm when looking at the most extensive variant, the SentiWordNet variant.

The precision and recall of the prior polarity classifiers however are better than those of the AdaBoost approach using 50 weak learners and only tags. This indicates that even though the accuracy is lower, the prior polarity classifiers compete with the AdaBoost classifier on polarity detection.

RBEM – For the RBEM algorithm we now let negative patterns emit negative sentiment instead of positive sentiment as we did for subjectivity detection. We also take the bipolar class into account by saying that a message is bipolar if both positive and negative patterns match the message.

The left half of the mid section of Table 17 shows the mean scores for each metric. Even when we count the misclassification of subjective messages we miss out due to insufficient labeling, the accuracy is already comparable to the highest accuracy for AdaBoost and higher than that of the prior polarity classifiers. When we disregard subjective messages for which no patterns are present in our model, we obtain the best accuracy of all approaches tested, 83.9%. This is an improvement of 33.7% over the majority class guess.

The precision and especially recall of the RBEM algorithm are much higher than those of other approaches. The RBEM algorithm is thus the most favored approach by the experiments conducted.

To investigate how much we can increase the performance of the RBEM algorithm, we investigate how much more the accuracy increases when we have more patterns. Recall that the number of patterns we have in our models that show the scores shown in Table 17 are shown in Table 8. We label more patterns as described in Section 2.2.3 to get the number of patterns shown in Table 15. We now have 81 more patterns than we had previously. Adding these 81 additional patterns was done in approximately six hours of dedicated labeling. A linguist however would most likely do this much quicker. We mainly label more on positive and negative patterns as we expect to gain the most with these pattern types. Moreover, as our Dutch model was relatively small with respect to our English model, we mainly label Dutch patterns. The scores for our metrics we then get for the RBEM algorithm are shown in Table 16. The percentage of polar messages that we do not manage to find due to insufficient labeling drops from 12.9% to 9.4%. The accuracy increases 72.4% to 74.1% when taking all messages into account. When we leave out the messages we cannot find due to insufficient labeling, our accuracy increases from 83.9% to 84.2%, indicating that the newly labeled patterns not only allow us to classify those messages we could not classify previously but also help correct the classification of messages that we misclassified previously.

Conclusions on Language Identification, Subjectivity and Polarity Detection
For language identification, we compared LIGA against a state-of-the-art approach. LIGA consistently outperforms this approach in all experiments including different sizes of labeled data and specialization

Table 15: The number of patterns present in the English and Dutch models of the RBEM algorithm after more labeling.

English		Dutch	
Type	Count	Type	Count
Amplifiers	67	Amplifiers	29
Attenuators	12	Attenuators	3
Rightflips	39	Rightflips	7
Continuators	10	Continuators	4
Leftflips	5	Leftflips	2
Negatives	541	Negatives	339
Positives	308	Positives	177
Stops	0	Stops	2

Table 16: The performance scores for the RBEM algorithm with the model of Table 15.

With Missed	A	0.741
	P	0.733
	R	0.876
Without Missed	A	0.842
	P	0.832
	R	0.955
Missed %		9.4%

and generalization experiments. We compared the AdaBoost algorithm for subjectivity detection and the RBEM algorithm for polarity detection against other competitor approaches and baselines. The experiments favor both AdaBoost algorithm and the RBEM algorithm with respect to accuracy, precision and recall.

The results are summarized in Table 18 where for each approach we experimented with only the setting which showed the highest accuracy (note the highest F-measure) is listed. Missed out messages are included in the accuracies and hence these are lower than we would have if we had sufficiently labeled data. For the prior polarity classifiers we only list using SentiWordNet as the results of the other settings are too heavily biased due to the number of messages missed out. For results on POS-tagging we refer to [Schmid, 1994]. The conclusions resulting from the experiments conducted achieve **Goal 1** presented in Section 3, i.e. we justify that using our presented approaches for each of the four steps is favored when we compare each approach against other competitor or baseline approaches. Note that for polarity detection however, the difference in using our RBEM algorithm and using AdaBoost is extremely small by Table 18. Labeling more patterns for the RBEM algorithm will increase this difference.

3.3.2 Complete Process Evaluation

In Section 3.3.1 we show that for each step individually, our algorithms outperform other algorithms we compared against. These results however may only hold locally. It may be the case that even though we use a proven good solution for a subproblem, leaving this subproblem out yields better results as perhaps the small error that is introduced, only introduces additional noise. We investigate this by looking at the benefit each step has on the accuracy of the sentiment analysis problem, thus achieving **Goal 2**.

We measure the impact each step has on the complete process by comparing two settings repeatedly. We first evaluate the complete process containing all four steps on the test set described in 3.2.5. This yields the gold standard to which we compare another setting where one of the four steps is left out, as is shown in Figure 12 for the subjectivity detection case. Each step is left out once. By leaving out a single step and comparing it to the case when the step is included, we analyze the addition of this step has on solving the sentiment analysis problem. Note that leaving out a step only influences succeeding steps. Whenever we leave out a step, we need some heuristics to cope for not having that information. The heuristics we use when leaving out each step individually are as follows. Remember that leaving out a single step will result in either of the three more generic cases we presented when describing the motivation for the complete process experiments in Section 3.1.1.

Table 17: The overall accuracy (A), precision (P) and recall (R) of algorithms for polarity detection. The instances per algorithm having the highest F-measure are shown in bold. The 'Missed %' row lists the fraction of subjective texts not found as such due to insufficient labeling.

Naive Bayes	All	MI 2000	MI 4000	MI 8000
	A 0.616	A 0.504	A 0.506	A 0.511
Tokens	**P 0.551**	P 0.459	P 0.460	P 0.467
	R 0.393	R 0.297	R 0.301	R 0.313
	A 0.585	A 0.491	A 0.493	A 0.482
Tags	P 0.543	P 0.439	P 0.440	P 0.427
	R 0.386	R 0.281	R 0.287	R 0.259
	A 0.589	A 0.495	A 0.494	A 0.502
Mixture	P 0.546	P 0.443	P 0.441	P 0.453
	R 0.387	R 0.286	R 0.287	R 0.292
	A 0.545	A 0.502	A 0.489	A 0.510
Patterns	P 0.497	P 0.419	P 0.447	P 0.428
	R 0.338	R 0.313	R 0.302	R 0.326

SVM	All	MI 2000	MI 4000	MI 8000
	A 0.656	A 0.504	A 0.507	A 0.420
Tokens	P 0.795	P 0.500	P 0.688	P 0.675
	R 0.431	R 0.013	R 0.023	R 0.478
	A 0.451	A 0.551	A 0.531	A 0.544
Tags	P 0.559	P 0.549	P 0.523	P 0.533
	R 0.532	R 0.490	R 0.473	R 0.452
	A 0.654	A 0.540	A 0.547	A 0.564
Mixture	P 0.699	P 0.542	P 0.557	P 0.555
	R 0.486	R 0.496	R 0.499	R 0.464
	A 0.637	A 0.500	A 0.503	A 0.514
Patterns	**P 0.646**	P 0.500	P 0.542	P 0.575
	R 0.564	R 0.131	R 0.261	R 0.279

	RBEM	Prior, Tok	Prior, Tok+Tags	Prior, SWN
	A 0.724	A 0.602	A 0.552	A 0.681
With missed	P 0.719	P 0.611	P 0.577	P 0.737
	R 0.862	R 0.583	R 0.550	R 0.498
	A 0.839	**A 0.769**	A 0.778	A 0.687
Without missed	**P 0.828**	**P 0.785**	P 0.831	P 0.741
	R 0.953	**R 0.747**	R 0.664	R 0.712
Missed %	12.9%	21.4%	28.7%	0.9%

AdaBoost	25 models	50 models	75 models	100 models	250 models
	A 0.664	A 0.691	A 0.685	A 0.678	A 0.698
Tokens	P 0.667	P 0.707	P 0.692	P 0.686	P 0.706
	R 0.638	R 0.654	R 0.649	R 0.644	R 0.658
	A 0.699	**A 0.723**	A 0.715	A 0.703	A 0.691
Tags	P 0.704	**P 0.729**	P 0.722	P 0.709	P 0.697
	R 0.668	**R 0.691**	R 0.685	R 0.677	R 0.666

Table 18: A summary of the single step experiments. Baseline approaches are shown in red, direct competitors of our used approaches are shown in blue and the approaches we use are shown in **bold**.

Step	Approach	Accuracy
Language Identification	N-gram using trigrams and N-gram frequencies, trained on 50% of the data	93.1%
	LIGA using trigrams, trained on 50% of the data	**97.5%**
Subjectivity Detection	Majority class guess	59.9%
	Naive Bayes using the top 8000 POS-tags based on MI	67.0%
	SVM using all patterns	74.1%
	RBEM missing out 8.7% of subjective messages	62.4%
	Prior polarity classifier using SWN	57.9%
	AdaBoost using POS-tags and 50 weak learners	**76.2%**
Polarity Detection	Majority class guess	50.2%
	Naive Bayes using all POS-tags	61.6%
	SVM using all patterns	63.7%
	RBEM missing out 12.9% of subjective messages	**72.4%**
	Prior polarity classifier using SWN	68.1%
	AdaBoost using POS-tags and 50 weak learners	72.3%

- **Leaving out Language Identification** – When we do not know the language of a message, we can no longer use language-specific models and hence need to apply more generic models. We thus need to combine the language-specific models into one. For POS-tagging we simply apply all models that are present. We do assume that we are interested in English and Dutch texts and hence we only apply these two models. As the POS-tagger assigns a probability of certainty with each assigned tag, we can sum up these probabilities and apply that model of which this sum is the greatest. For subjectivity detection we apply both the English as well as the Dutch model on the message, sum up the scores for both languages and assign the resulting class. For polarity detection we take the same approach as for subjectivity detection.
- **Leaving out POS-tagging** – Both subjectivity detection as well as polarity detection use POS-tags as features. When we do not have these tags we need to resort to using tokens only. For subjectivity detection this means we have to train AdaBoost using tokens only. For polarity detection, we use our patterns without regarding the POS-tags.
- **Leaving out Subjectivity Detection** – If we do not know whether a text is subjective or not, we need to determine this in the polarity detection step, combining subjectivity detection and polarity detection into one. There are now two approaches we can take. Firstly, we can say that whenever we have a 0 polarity score, the message is objective. In this case however, we rule out subjective texts that are not polar, over-classifying the objective messages. Secondly, we can say that a message is objective if no patterns could be matched against it. In this case however, we label a lot of objective messages containing polar patterns as subjective, over-classifying subjective messages. We apply both heuristics but it turns out that there is no difference in their performances, indicating that our test set does not contain objective messages containing private state nor subjective messages that are not polar. The lack of these types of messages is explained by the relatively small size of our test set as well as that we only included obvious cases to not bias the test set whereas these types of messages are traditionally closer to the decision boundary. Our results on the two different heuristics are hence not conclusive, which is fine as this is not our intention.
- **Leaving out Polarity Detection** – When we leave out polarity detection we in fact cannot determine the polarity of a message and hence we cannot deliver the required output. To overcome this problem, we can use AdaBoost to combine subjectivity detection with polarity detection. This results in testing the applicability of AdaBoost on polarity detection. Figure 11 shows how this conceptually looks like, yielding a three-way classification instead of a two-step binary classification. Though the applicability of AdaBoost on polarity detection is also presented in Section 3.3.1, we analyze the effect it has on the complete process by using it as a three-way classifier here.

Table 20 shows the accuracies on the test set for all different scenarios. The columns list the steps left out whereas the rows list the accuracies after each step. The accuracy for a single step is computed by dividing the number of messages correctly classified by that step by the number of messages correctly

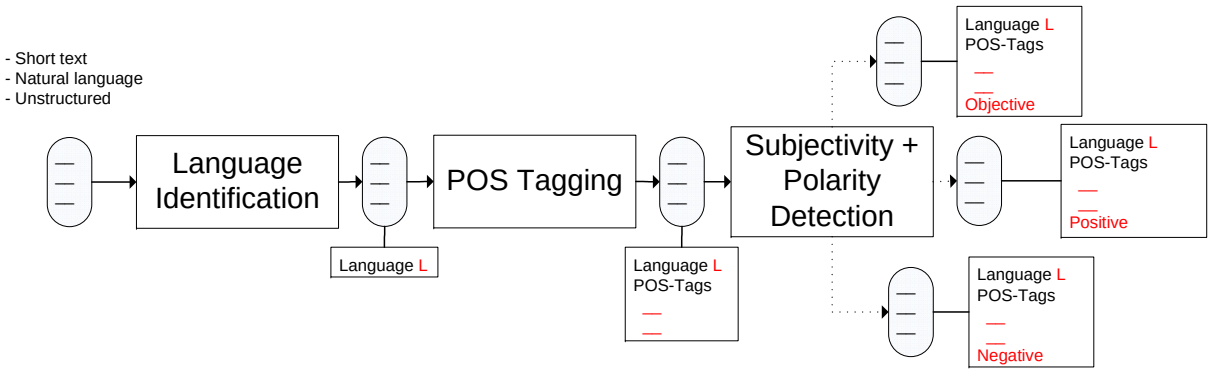


Figure 11: Leaving out polarity detection means we have to do a three-way classification in the last step. Note that for this experiment, we only pass on correctly classified instances to the next step.

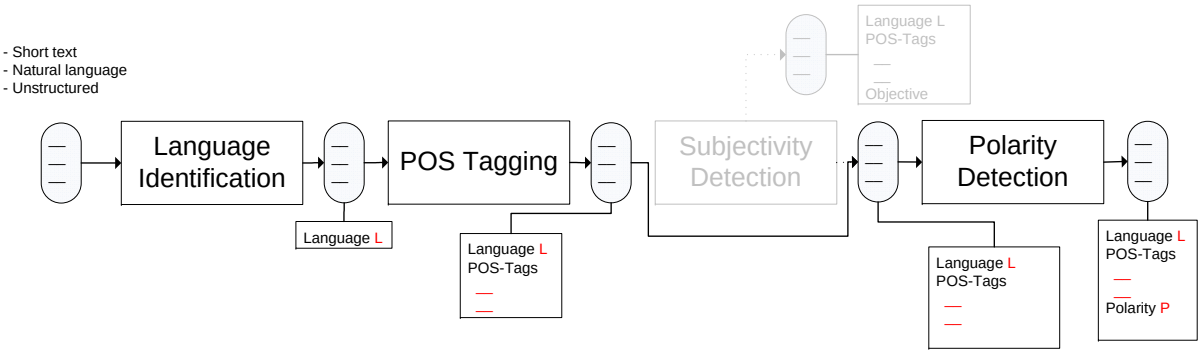


Figure 12: Leaving out a single step (subjectivity detection in this case) from the complete process to measure the addition.

classified up to that step. The last row indicates the number of messages that are subjective but which our polarity detection step could not classify as such due to insufficiently labeled data. The *complete* row lists the accuracies on the entire test set, computed by dividing the number of messages correctly classified by all steps by the corpus' size, 120 messages. As POS-tagging is merely used to expand our feature space, we do not evaluate the accuracy after performing this step.

As we can see from the last column, the final accuracy when not leaving out any step on the test set is 69.2%. Table 19 shows the number of instances correctly and incorrectly classified by each of the steps when not leaving out any step. Note that in this experiment, for the evaluation purposes, only correctly classified instances are passed on to the next step. Language identification is the most accurate step whereas subjectivity detection is the least accurate. If any improvement is to be made, subjectivity detection is the first place to start. Additionally, more extensive labeling for the RBEM algorithm could help as well as 7 out of the 10 misclassified instances are due to insufficient patterns present in our model.

Table 20 shows that leaving out polarity detection has the greatest impact on the complete process, this step is the most crucial one. Moreover, the accuracy of the polarity detection step is influenced greatly by the (non-)presence of the subjectivity detection as leaving out subjectivity detection significantly decreases the polarity detection's accuracy. Leaving out the language identification step or the POS-tagging step can be partially remedied by the subjectivity and polarity detection steps. Including them however does increase the accuracy, indicating their importance nonetheless. Note that leaving out language identification may also introduces errors in the POS-tagging algorithm as the wrong language model may be chosen.

Table 19: The number of instances (in)correctly classified by each step when not leaving out any step.

Step	Incorrect	Correct
Language Identification	1	119
Subjectivity Detection	26	93
Polarity Detection	10	83

Conclusion
Each step is beneficial to include in the complete process as leaving one of them out results in worse accuracies. As steps performed earlier in the process are actually used as pre-processing for later steps, leaving out later steps has a greater impact since more specific classifications are made. The overall accuracy of the complete process on our test set is 69.2%, which would also be the expected accuracy in operational settings given the same type of data we have in our test set. With these observations we achieve **Goal 2** presented in Section 3.

Table 20: The correctness scores after each step (vertical) for leaving out each possible step (horizontal).

	No LI	No POS-tag.	No Subj. Det.	3-Way Pol. Det.	**All included**
Acc. of LI	–	0.992	0.992	0.992	0.992
Acc. of Subj	0.692	0.689	–	0.782	0.782
Acc. of Pol	0.880	0.829	0.513	0.538	0.892
Acc. of Complete	0.608	0.567	0.508	0.417	**0.692**
% Missed in Pol. Det.	0.800	0.429	0.810	–	0.700

3.3.3 Error Propagation Experiments

In Section 3.3.2 we show the importance of each step on the complete process. To quantify the propagation and influence of errors made during each step, achieving **Goal 3**, we deliberately introduce label errors at each step and compare the result against not having any errors. The corpus on which we do this needs to be as accurate as possible to avoid introducing random corrections instead of errors. To this end we use the test set presented in Section 3.2.5.

For each step, we introduce errors that can be distinguished into two types; singular errors and composite errors. The former are errors of one kind, caused by one step only. The latter are errors of multiple kinds, caused by more than one step. This way we can not only quantify what it means to have each single step as accurate as possible but also what it means to have any combination of steps as accurate as possible.

What we in fact measure is the following. Let the errors introduced in step A be denoted *Err(A)*. Then what is measured is the conditional error of a step B occurring after step A, denoted *Err(B | Err(A))*. Note that when we deal with composite errors, we measure conditional errors of the form *Err(B | Err($A_1, .., A_n$))*. This way we quantify the error propagation of a step as well as combinations of steps.

To make sure we introduce exactly the amount of errors required, we replace the process up to and including the step in which errors are to be applied with the test set, simulating a completely correct process up to a certain point if no errors are introduced. In this test set we then introduce errors and provide it as input for the next step. This process is shown schematically in Figure 13 where we investigate the error propagation of only the language identification step. We use the test set as a replacement of the language identification step and introduce a fixed fraction of errors (20% in this particular example, resulting in an 80% accurate test set) in the labels of the test set.

When performing any step, only errors introduced at previous steps are of any influence. As polarity detection is the last step of the process, introducing errors at this step does not influence any succeeding steps but rather the outcome directly. It is of little use to investigate the propagation of errors made in polarity detection as there is nothing to propagate to. The errors we introduce are constructed as follows.

- **Language identification errors** – These errors are introduced by simply flipping the label assigned to the message in the construction of the test set from Dutch to English or vice versa.
- **Part of speech tagging errors** – These errors we introduce by swapping each POS-tag with a 50% chance with any other uniformly chosen part of speech tag. We thus only introduce errors in the ordering of the tags, not in the (non-)presence of the tags.
- **Subjectivity detection errors** – These errors are introduced in the same way language identification errors are, by flipping the label.

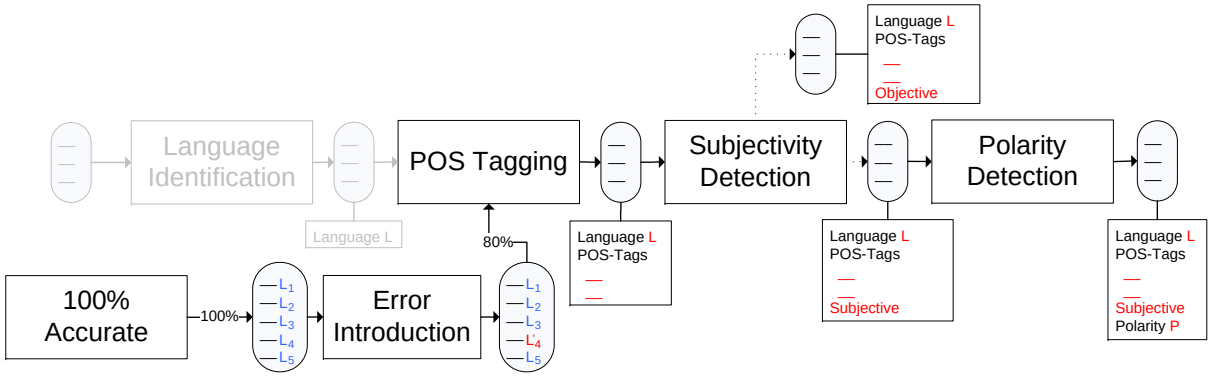


Figure 13: Providing an input in which we know a number of errors exist (20% in this case). One or more steps are taken out of the process and replaced by the test set.

In all cases we determine a fixed number of errors that need to be introduced. We randomly sample the number of messages required and introduce the errors of the type we are investigating into each of these messages. In all cases, the accuracies we report are the mean over 10 runs for a given percentage of errors. In each run we resample the messages in which errors are introduced. We perform this process from 100% correct (no errors) to 0% correct (all messages contain errors) with steps of 5%.

With all subfigures of Figure 15 we show a dashed line titled *4 Step*. This line shows the constant 0.692, corresponding to 69.2%. This is the accuracy of our entire four-step process as presented in Section 3.3.2 on the test set. The error propagation will always start with an accuracy above this line as we replace one or more of our four steps with a (more) accurate data set in which we do not introduce any errors at first. We thus have a correct data set for our error propagation experiments where our four-step process has a step that introduces at least one error as was shown in Section 3.3.2.

Figure 15(a) shows the influence of errors on the accuracy of the POS-tagging. The only errors that can be introduced before POS-tagging are language identification errors. We thus only measure *Err(POS | Err(LI))*. The accuracy of the POS-tagging step is determined by dividing the number of POS-tags that are correctly assigned by the total number of POS-tags present. The accuracy of the POS-tagging given perfect language identification is 87%. Increasing the number of language identification errors further decreases the accuracy of the POS-tagging step, resulting in a completely wrong tagging when no language is identified correctly. This can be explained by the fact that Dutch and English share no POS-tags in common.

Figure 15(b) shows the accuracies of the subjectivity detection step when errors have been introduced in previous steps. Having POS-tagging errors hardly influence subjectivity detection. This is due to the way our AdaBoost approach operates, only looking at presence of POS-tags and incorporating lexicons on top of POS-tags. Having both language identification errors as well as POS-tagging errors yields similar accuracies to just having language identification errors. We can thus say that *Err(Subj | Err(LI, POS))* is comparable to *Err(Subj | Err(LI))*.

Figures 15(c) and 15(d) show the accuracies of the polarity detection step when singular and composite errors have been introduced respectively in previous steps. As the output of the complete process coincides with that of the polarity detection step, measuring the accuracy after the polarity detection step coincides with measuring the accuracy of the complete process. It is remarkable that having just language identification errors results in lower accuracies than having composite errors including language identification errors. This explanation could be due to many factors. Two of which are that additional errors on top of language identification errors have a remedying effect or even just due to randomness.

From Figure 15(d) we see that all composite errors encompassing subjectivity detection have similar effect on the accuracy of polarity detection. This effect is less drastic than the composite errors not encompassing subjectivity detection. It turns out that errors made in the subjectivity detection step remedy errors made in language identification or POS-tagging. In contrast to what we see with subjectivity detection, POS-tagging errors do greatly influence polarity detection. This is because the patterns we have for

the RBEM algorithm for polarity detection rely on POS-tags to disambiguate words. If we do not have such POS-tags, all positive and negative patterns present in our RBEM model are in fact prior polarity patterns. These prior polarity patterns can comprise more than one word in an attempt to disambiguate words but in that case we are using our RBEM algorithm to also perform an encapsulated POS-tagger, something we do not do for these experiments.

In Figures 15(c) and 15(d) we also show accuracies of three baselines, a Naive Bayes, an SVM and a prior polarity baseline. These baselines are three different approaches that perform sentiment analysis in one step without any prior knowledge. We hence do not know the language of a message nor have its POS-tags available. The process that belongs to these baselines is thus a three-way classification as shown in Figure 14. We show these baselines as when our four-step process is performing worse than either of them, it no longer makes sense to use such a four step process. The error propagation thus shown in Figures 15(c) and 15(d) that results in accuracies lower than the baselines is not interesting as we could rather replace our process with these baselines.

For singular errors, the prior polarity baseline performs better whenever 20% or more errors have been introduced except for the POS-tagging errors of which the boundary is with 40% or more errors. For composite errors, having language identification and POS-tagging errors performs worse than the prior polarity baseline from 35% errors or more. For language identification, POS-tagging and subjectivity detection errors combined, the boundary is with 40% or more. For the other error combinations, the boundary is with 50% errors or more, we can thus afford having more than half wrong before performing worse than the prior polarity baseline. The Naive Bayes and SVM baselines perform worse than the prior polarity baseline and hence we can have more errors to still perform better than these baselines.

The performance of the Naive Bayes and SVM baselines being worse than a prior polarity classifier may seem suspicious. We briefly investigate how this is possible. First of all, as we now have a three-way classifier. A majority class guess would always result in a most 33.3% accuracy as our test set contains three classes which are each equally represented. The Naive Bayes and SVM classifier hence do perform better than the majority class guess. The discrepancy of class representation in the training set and our test set is why the Naive Bayes and SVM classifier, trained on our training set, perform this bad. Moreover, when applying a Naive Bayes or SVM classifier on the complete three-way classification, using POS-tags – in contrast to what we saw with the single step experiments in Section 3.3.1 – significantly increases accuracy. However, we do not have any prior knowledge and hence cannot utilize POS-tags. Our test set also contains little vague cases to ensure its integrity. A prior polarity classifier is strong in detection obvious cases but performs terrible in vague cases as there the prior polarity of a word often is not the actual polarity.

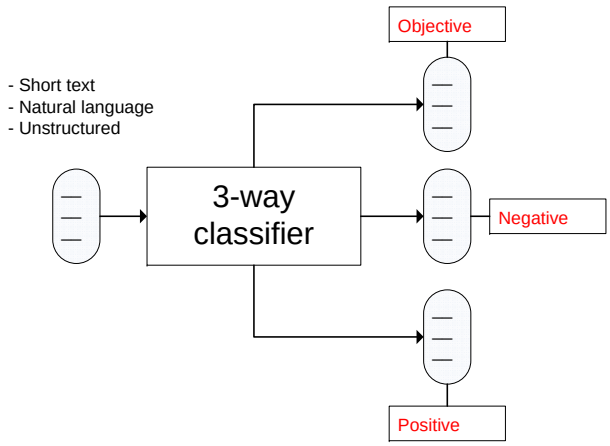
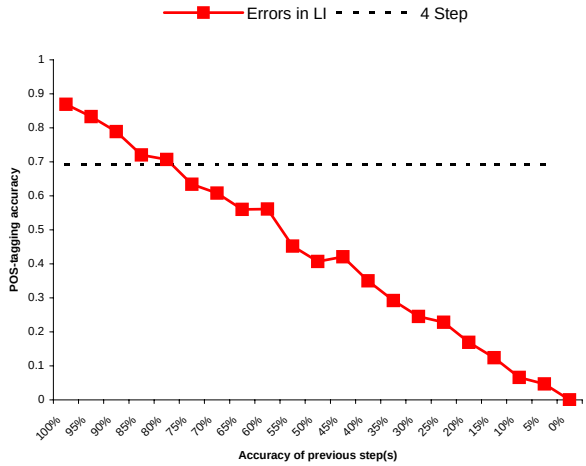


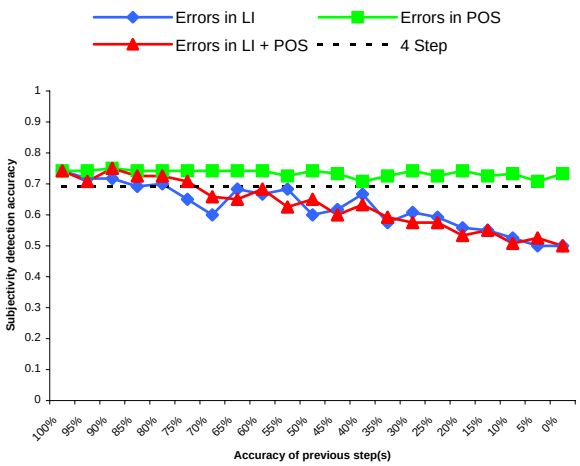
Figure 14: Process for the baselines performing sentiment analysis in one step.

Conclusion

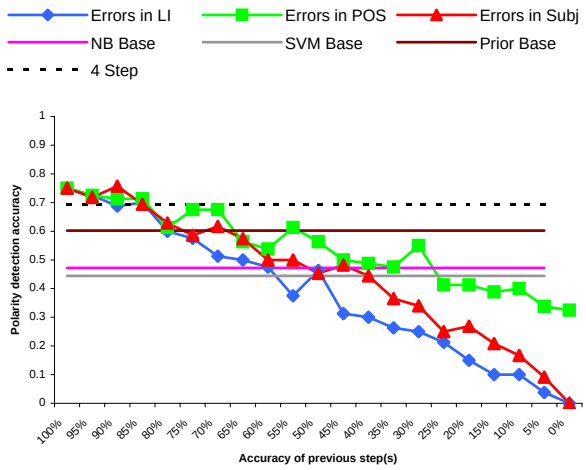
We quantified the effect of any combination of errors introduced at each of the steps. Errors made during the language identification most drastically influence succeeding steps whereas POS-tagging errors least affect succeeding steps. The former can be due to that succeeding steps make use of language-specific models and approaches. The latter can be due to both the type of POS-tagging errors we introduced,



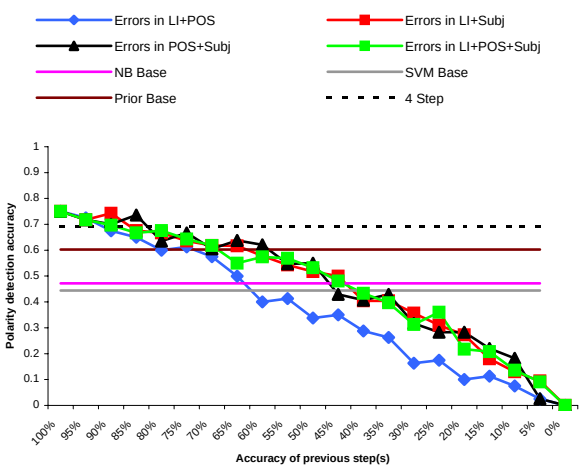
(a) Accuracy of POS-tagging given the accuracy of language identification after introducing errors



(b) Accuracy of subjectivity detection given the accuracy of previous steps after introducing errors



(c) Accuracy of polarity detection given the accuracy of previous steps after introducing errors in one step



(d) Accuracy of polarity detection given the accuracy of previous steps after introducing errors in multiple steps

Figure 15: The accuracy of POS-tagging 15(a) given errors in language identification. The accuracy of subjectivity detection 15(b) when errors are introduced in previous step(s). Also shown are the accuracies of polarity detection, which coincides with the complete process, when singular 15(c) or composite 15(d) errors have been introduced in the previous step(s). For the complete process, three baselines solving the complete process at once are shown.

only errors in the order of tags, as well as due to the way our subjectivity detection step works, mainly regarding presence of POS-tags but also the use of POS-tag independent word lists. Through these experiments we achieve **Goal 3**.

By looking at Figure 15, we can identify the critical points for errors made at each step where it is no longer justified to use our four-step approach as a more generic one-step approach then performs better. When we look at Figure 15(c) we see the effect of singular errors on our complete process. For language identification, we can have at most 25% errors as having more errors results in worse performance than the prior polarity baseline approach. For POS-tagging we can have at most 35% errors. For subjectivity detection it is not entirely clear as it drops below the threshold of the prior polarity classifier at an error rate of 25% but goes over it again having 30% errors and then stays below the threshold. To be on the safe side, we can say that we can have an error rate of at most 25%. If the error rate grows larger than any of these thresholds for the step belonging to that threshold, it is no longer justified to use our four-step approach.

3.4 Traditional Survey Experiments

We illustrate and investigate the inherent difficulties and possible solutions thereof of making a mapping between our automated sentiment analysis results and the results of a traditional survey which allows us to achieve **Goal 4** presented in Section 3. To this end we use the crawled and survey data described in Section 3.2.1. We attempt to align the results of the survey with those of our automated sentiment analysis.

The alignment between our sentiment analysis and the survey is performed at two levels of granularity; global and segment-based and on two different data levels; natural language for our sentiment analysis against score-based ratings for the survey and natural language for both. When making a segment-based alignment, we take age into account to make a segmentation. The only social medium that allows us to extract age information is Hyves. We thus use only Hyves for our segment-based alignment whereas we make an alignment with all three social media in the global alignments. Moreover, since the survey is held among a representative population of Dutch society, only Dutch opinions are expressed in the survey. We thus use only the Dutch social media messages in our sentiment analysis.

We first define a methodology to align the sentiment analysis' results with those of the traditional survey in Section 3.4.1 and identify the problems that arise with it. We then first make an alignment using natural language for both our sentiment analysis as well as the traditional survey in Section 3.4.2. We additionally investigate an alignment using natural language for our sentiment analysis against using the score-based ratings for the survey in Section 3.4.3. A summary of our findings is given in Section 3.4.4.

3.4.1 Alignment Methodology

When making a mapping between our sentiment analysis and a traditional survey, we run into inherent problems between the two. We explicitly list each of these problems with a detailed elaboration on what it implies and how it can be solved if possible. Some of these problems we can easily overcome but some even require a study on its own. Note that the type of study required to solve all of these problems is not a computer science study but rather a sociological study. We hence do not attempt to exhaustively solve all these problems but rather propose potential solutions and show whether or not these solve the problem or not.

Different data sources – Traditional surveys typically rely on number based scorings to express sentiment on a certain entity. On social media however, we do not have such scores but rather have natural language expressing sentiment. Not only does this give difficulties in creating a mapping between the two, it also presents an inherent problem. The way people express sentiment on request and in a scoring fashion on a predefined scale is expected to differ from the way people do this in voluntarily fashion being much broader and free using natural language.

Even if we use natural language for both our sentiment analysis as well as our traditional survey, we still have a data source problem. The way people use language and more importantly, the way they express sentiment in natural language, also differs when people either voluntarily share it or when they are pro-actively asked for it. Sentiment expressed on social media is voluntarily given by people and hence implies the person has a personal stance, otherwise this person would not have shared it. In traditional surveying, people are triggered to think on entities they perhaps have no strong personal stance on. This is a typical difference in the two data sources we can not overcome.

The difference in the data sources thus lies with the way in which the data is constructed. Exactly identifying what these differences are and finding solutions to overcome them is out of the scope of this project as it would be a study on its own. Such a study moreover focuses more on sociological science rather than computer science.

Differences in segment – It is very likely that people of one age have a different opinion than opinion of another age. It hence makes sense to segment our results based on age groups. We thus to make a segmentation into age groups for the segment-based experiments. As the survey is filled in by a representative sample of the Dutch society, we use its distribution in age to create approximately equal-height age groups. The resulting groups and the distribution are shown in Figure 16(a). When we partition our sentiment analysis results obtained from Hyves into the same age groups, we get the distribution shown in Figure 16(b). The clear mismatch between Figures 16(a) and 16(b) motivates am alignment on the segment level rather than globally.

Amount of data – The amount of opinions resulting from Hyves may be too little. This may be due to the model used for our polarity detection algorithm containing too little patterns to find sufficient data or due to the little timespan in which we scraped Hyves, being only three days. The number of messages on politicians is higher than the number of messages on the other entities though.

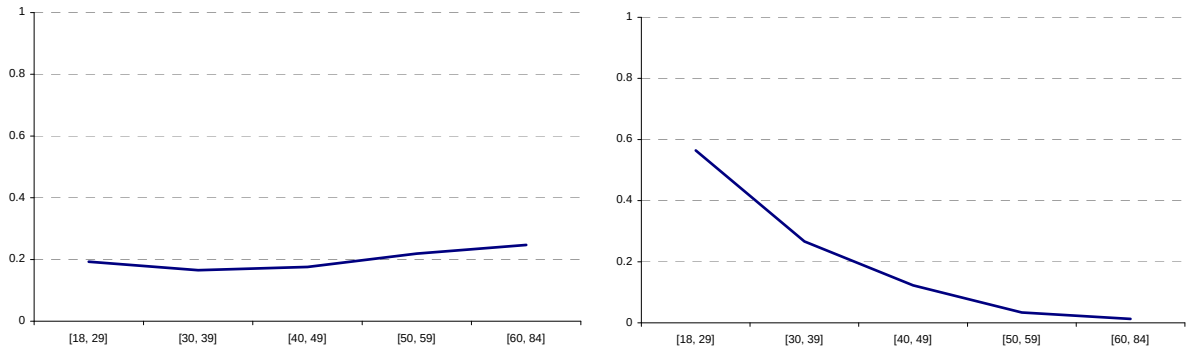


Figure 16: The age groups of approximately equal height for the survey 16(a) and the same groups for Hyves 16(b). The resulting groups are {[18, 29],[30, 39],[40, 49],[50, 59],[60, 84]}.

3.4.2 Natural Language Alignment

In this experiment, we regard the results from the traditional survey as a ground truth using the open field data set described in Section 3.2.1. We moreover assume to have a ground truth dataset containing sentiment from social media, this is the ground truth dataset described in Section 3.2.2. We assume that a mapping between the traditional survey's results and those of our sentiment analysis exist and that our traditional survey indeed is the ground truth. The traditional survey allows respondents to fill in an open 'comment' field to support their answer. This field contains natural language text and possibly further explains the opinion of the respondent. As this field contains natural language, we can more naturally map it onto our sentiment analysis which is also performed on natural language, be it from a different source.

The setup of our alignment is shown in Figure 17. Even though we identified problems with making a mapping between the sentiment of our social media and the sentiment of our survey in Section 3.4.1, we

now assume that such a mapping exists although we do not explicitly know it. The existence of such a mapping allows us to compare metrics on both datasets, for example accuracy. The two metrics resulting from the mapping we assume to exist we will look at are accuracy and the distribution of sentiment.

As we have two ground truth datasets, we can investigate whether our sentiment analysis is accurate enough to find the mapping we assume to exist. Additionally, we can investigate what will happen if we introduce errors in our social media's ground truth dataset and quantify how well our sentiment analysis will maintain the mapping. This quantification is similar to what we do with the error propagation experiments in Section 3.3.3.

Even though our two ground truth datasets are both on natural language, they still result from two different source; social media and a traditional survey. We investigate whether or not these two data sources are comparable by running our sentiment analysis on both and comparing the obtained results.

We could also assume a mapping between natural language sentiment from our social media and the score-based ratings from the traditional survey. We however do not perform experiments assuming such a mapping. Figure 17 also shows a mapping arrow between the natural language sentiment of the traditional survey and the score-based ratings of the survey, mentioning that it is not likely that such a mapping exists. If the mapping would exist, it would be interesting to actually investigate using the score-based ratings. This however is not likely due to three reasons.

- The **Different data sources** problem described in Section 3.4.1 states the inherent difference between sentiment expressed in natural language and sentiment expressed in a scoring-based fashion.
- During our manual labeling of the open field responses we found that the sentiment expressed in natural language relatively often disagrees with the score given. An example hereof is someone giving 'just' a neutral score for an entity whereas the corresponding open field response is *I have good experience with this product*, indicating positive sentiment.
- In Section 3.4.3 we will not assume a mapping between the two but rather try out possible mappings between the score-based ratings of our survey and the natural language of our social media. We show that there is a lack of similarity, indicating that the mapping used is not correct.

Sentiment Distribution Evaluation

As a metric produced by our mapping we regard the distribution of sentiment in this experiment. We run our four-step sentiment analysis approach on the ground truth set described in Section 3.2.2. As we have seen in Section 3.4.1, the [18, 29] age group is overly represented on Hyves. We assume that this also holds for the other social media – of which we do not know the age distribution as we cannot request age of a use – and hence we use **Sample 2** described in Section 3.2.1 rather than **Sample 1** to reduce the noise in our data since we have more data. This sample dataset is also assumed to be ground truth.

As both of our datasets, one from social media and one from the traditional survey, are 100% accurate for we assumed them to be ground truth, we can use their sentiment distributions as a reference point. We do this on a per-entity base. The sentiment distributions of our ground truth set for each entity are shown in Figure 24. The sentiment distributions for our **Sample 2** set on the same eight entities are shown in Figure 25. There is not much similarity between the distributions of the two data sources but we assume that what we see here is a result of our mapping and hence these distributions are our reference point as they are based on ground truth.

In order to make our comparison between the distributions more crisp, we introduce a metric to express the difference in sentiment distributions. A metric that is often used to express difference in machine learning tasks is the *mean squared error* (MSE). We have no way to find out whether or not this metric is also suitable for our purposes but use it for its common use. A more detailed, sociologically oriented study could perhaps find more expressive metrics for the difference in distributions. We denote the fraction of positive sentiment of a sentiment distribution d_i on an entity e_j as $f_{i_p}(e_j)$. Similarly we denote the fraction of negative sentiment on an entity e_j as $f_{i_n}(e_j)$ and the fraction of objective sentiment on an entity e_j as $f_{i_o}(e_j)$. The MSE of two sentiment distributions d_1 and d_2 on entities $e_1, .., e_n$ is now defined

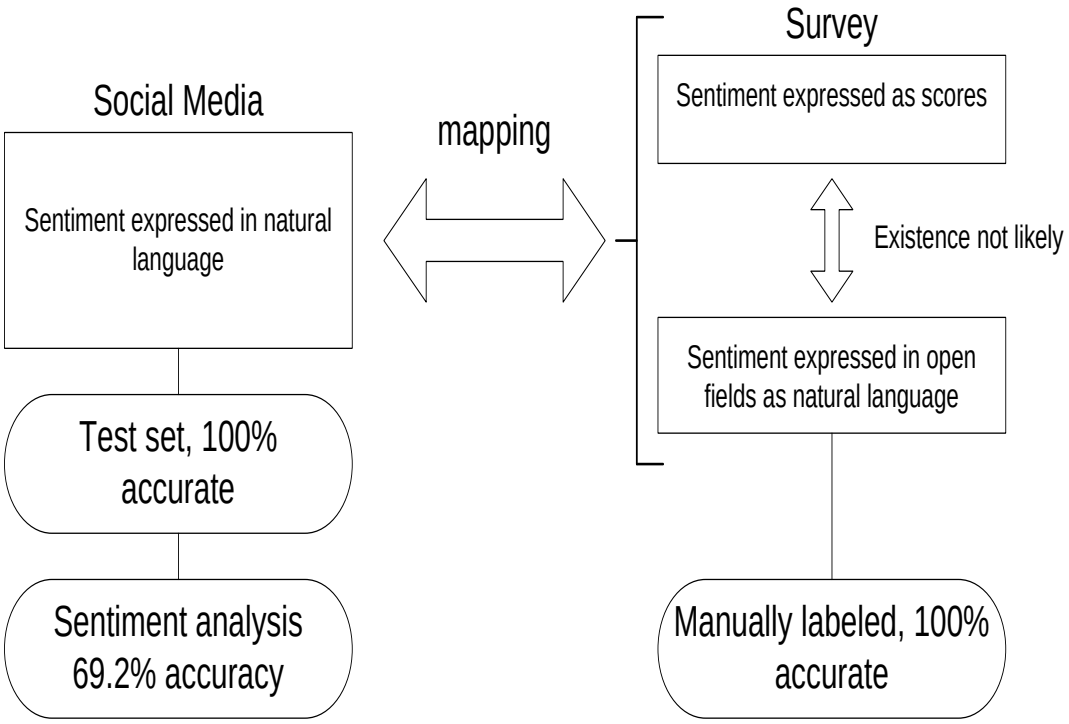


Figure 17: The setup for our natural language-based alignment. We have a manually labeled, 100% accurate dataset which is our test set described in Section 3.2.5. We assume that a mapping between our data exists, even though we do not know it explicitly. The open field responses of our survey are also manually labeled and assumed to be 100% accurate.

in Equation 7. The MSE for our reference point on 100% accurate datasets is 0.098.

$$MSE(d_1, d_2) = \frac{\sum_{i=1}^{n}(f_{1_p}(e_i) - f_{2_p}(e_i))^2 + (f_{1_n}(e_i) - f_{2_n}(e_i))^2 + (f_{1_o}(e_i) - f_{2_o}(e_i))^2}{3n} \tag{7}$$

We now perform our sentiment analysis on the ground truth set described in Section 3.2.2, the same data the distributions in Figure 24 are based on. We note that the obtained accuracy on the ground truth set is 69.0% which is very close to the 69.2% accuracy we found in Section 3.3.2 on our test set, further strengthening this earlier result. When performing our sentiment analysis on the ground truth set, we obtain the distributions shown in Figure 26. The sentiment distributions obtained after running our sentiment analysis on the ground truth set – resulting from a 69.0% accurate classification – do not really resemble the distributions of the 100% accurate ground truth set shown in Figure 24 when we look at the figures. A general observation is that our sentiment analysis tends to favor the objective class. This is because the model for our RBEM algorithm may contain too little patterns.

The MSE of the distributions shown in Figures 26 and 25 is 0.166. With respect to our 100% accurate dataset, the MSE has almost doubled. We only see some weak indications that our sentiment analysis is strong enough to establish the mapping we assumed to exist through the little resemblance present in Figures 26 and 24. We next quantify how accurate we need to be to obtain strong evidence that a mapping can be established.

Distribution Quantification of Labeling Error

Similar to what we do with our error propagation experiments in Section 3.3.3, we quantify how accurate we need our sentiment analysis to be to obtain strong evidence that a mapping between our sentiment analysis' results and those of the traditional survey can be established. We do so by deliberately introducing errors in our ground truth data set described in Section 3.2.2 and comparing the resulting sentiment distributions on a per-entity base with the sentiment distributions of **Sample 2** resulting from the survey data as described in Section 3.2.1.

We introduce errors in a predefined fraction x of labels of our ground truth set. We do this by uniformly

taking a fraction of x messages for each entity e, yielding a set of messages e_x. For each message in e_x we have a label being either one of *positive, negative, objective*. There are hence two classes that are not equal to the label of our message. We again uniformly assign one of these incorrect classes as the new, erroneous label to our message, thus introducing an error. We repeat this process 200 times and take the mean distributions – given by the mean fractions of positive, negative and objective sentiment – of these runs. We introduce errors with steps of 5% and start with having 5% errors and end with 60% errors. We start at 5% as we have already seen the distribution when not having any errors at all in Figure 24.

The sentiment distributions having 5% to 20% of all messages wrongfully labeled are shown in Figure 27. The sentiment distributions having 25% to 40% of all messages wrongfully labeled are shown in Figure 28. Finally, the sentiment distributions having 45% to 60% of all messages wrongfully labeled are shown in Figure 29. What we see is a gradual decrease in the fraction of objective messages whereas the fractions of positive and negative messages increases until the classes become approximately equally well represented. This is not strange since we uniformly assign other labels. It is hence important to note that as the sentiment distributions having 30% of all messages wrongfully labeled, whose error rate most closely approximates the error rate of our sentiment analysis on the same data, do not closely resemble Figure 24, the errors made by our sentiment analysis process are not uniform. In fact, our sentiment analysis typically classifies polar messages as objective as the most common error, which we again can explain by the little amount of patterns present in our RBEM model.

In order to make more crisp statements about the differences in distributions, we compute the MSE, defined in Equation 7, for each of the distributions with a given error rate against the 100% accurate distributions of our traditional survey shown in Figure 25. These MSE scores are summarized in Table 21. As we can see, the MSE decreases as the error rate increases. This means that the sentiment distributions of our social media more closely resembles the sentiment distributions of our traditional survey as more errors are introduced. We assumed however that a mapping existed between the two data source and have set our reference point on the 100% accurate datasets in which we obtained an MSE of 0.098. As performing our sentiment analysis on the social media data achieves an accuracy of 69.0%, having an error rate of approximately 30%, yields an MSE of 0.166 is an increase instead of a decrease, we cannot compare the sentiment distributions of any of social media data with the given error rates with the sentiment distributions resulting from our sentiment analysis. This is due to the type of errors made in our sentiment analysis against the type of errors we deliberately introduce into our dataset.

Table 21: The MSE scores between sentiment distributions on social media having a given error rates and the sentiment distributions of the 100% accurate **Sample 2** set.

Error rate	MSE
5%	0.091
10%	0.085
15%	0.080
20%	0.074
25%	0.067
30%	0.061
35%	0.058
40%	0.054
45%	0.052
50%	0.047
55%	0.045
60%	0.043

Accuracy Evaluation

For this experiment we regard the accuracy of our sentiment analysis as a metric produced by our mapping. This way, we evaluate the data of two different sources on obtained accuracy. To evaluate the data, we run our sentiment analysis on **Sample 1** described in Section 3.2.1, a representative sample of Dutch society containing open field answers of a traditional survey that is manually labeled. In Section 3.3.2, our sentiment analysis showed a total accuracy of 69.2% on our test set. As our sample **Sample 1** only contains Dutch open field responses, whereas our accuracy of 69.2% is on both English and Dutch data, we need to obtain the accuracy of our sentiment analysis on only Dutch social media-originated data first.

We perform our complete four-step sentiment analysis approach on only the Dutch data of our test set containing half of the messages – 60 messages, 20 positive, 20 negative and 20 objective – of our total test set. When doing so, we get the correctness scores presented in Table 22. Note that as we know we only have Dutch data, we do not need our language identification step. In total we correctly classify 42 out of the 60 messages, implying an accuracy of 70.0%. We also restate our finding that our sentiment analysis process obtains an accuracy of 69.0% on the ground truth set.

We next perform our sentiment analysis on **Sample 1** which we regard as ground truth data. When doing so, we obtain the confusion matrix shown in Table 23. The accuracy resulting from this confusion matrix is 71.8% which is very close the 70.0% accuracy of performing our sentiment analysis on only the Dutch part of our test set and also close the 69.0% accuracy obtained after applying out sentiment analysis on the ground truth set. It is thus likely that the data extracted from social media is comparable to the open field responses of a traditional survey.

Table 22: The number of messages correctly classified at each step on just Dutch data from our test set.

Step	Errors
Language Identification	0
Subjectivity Detection	47
Polarity Detection	42

Table 23: Confusion matrix of applying sentiment analysis on ground truth data. The rows are the true labels, the columns the predictions.

	Positive	Negative	Objective
Positive	324	8	118
Negative	8	122	101
Objective	18	22	255

3.4.3 Score-Based Alignment

In this section we investigate making an alignment between sentiment expressed in natural language from our sentiment analysis against sentiment expressed in score-based ratings from our traditional survey by looking at specific mappings. We will attempt to map the results of our sentiment analysis onto the score-based results of our traditional survey at the segment-level, at the global level and we even try to learn a mapping on our traditional survey data which biases our results as we use our comparison data constructively to define the comparison itself.

In addition to the general problems when making an alignment we identified in Section 3.4.1, we now run into additional problems listed next.

Differing scales – Our survey rates entities on a scale from 1 (very positive) to 5 (very negative) whereas our sentiment analysis process results in a label being one of *positive, negative, objective*. In order to align the two, we need to define some mapping. If we want to map the sentiment analysis' interval onto that of the survey, we have to do a discretization of the grading feature. This is a problem on its own studied in literature [Dougherty et al., 1995, Butterworth et al., 2004]. This is not a trivial task as we expect that respondents somewhat forcibly answering survey questions in a rating fashion differ in their opinions with respect to people voluntarily sharing their opinions online in natural language. We hence deal with a problem in which two inherently different worlds are to be aligned.

Ideally, we would like to use some prior knowledge we have on the distributions of both the survey rating as well as the online rating people perform. Doing this however, requires us to use the traditional survey data, which biases our experiments as we use it constructively to define the mapping. We can thus not use prior knowledge and instead we will make the unrealistic assumption that for the sentiment analysis a score of 0 or an objective label is neutral, anything below 0 is negative and anything above 0 is positive. For the survey, a rating of 3 is neutral, 1 or 2 is positive and 4 or 5 is negative. This mapping is the most natural mapping and we use it for the segment-based and global experiments.

Training mapping – Using the traditional survey's data in our mapping biases our results. Not doing so however seems unrealistic as it is domain knowledge we can easily obtain. For this reason we also experiment with using the survey data of one domain to create a mapping for other domains in our biased alignment. To do so, we need to make the assumption that entities in different domains having the same rating scale are generally rated with a similar attitude. By this we mean that one domain is not generally looked at more pessimistic or optimistic than another domain. This assumption again is not realistic.

Mapping criteria – Another aspect that is important when making a mapping is *what* will be mapped. As we deal with three different classes of sentiment (positive, neutral/objective, negative), we map each entity on these three classes in two ways. The first is whether or not a direct mapping can be made between the survey results and the sentiment analysis results. By this we mean that the fraction of positive, negative or neutral sentiment on a given entity is approximately equal in the survey and the sentiment analysis results. The second way is to map rankings rather than a direct mapping. By this we mean that we rank the entities within a domain in order of positiveness. Here, positiveness is defined as $positiveness = \frac{fraction\ positive}{fraction\ negative}$. The higher the value of *positiveness*, the more positive an entity is. We map the rankings in a pairwise manner per domain.

In order to easily obtain insights in the (dis)similarity of the mappings we make, we perform visual comparisons of the corresponding aggregated sentiments. The figures we analyzed are available in the Appendix B. Here we summarize the main conclusions of this analysis.

Segment-Based Alignment

For a segment-based alignment, we take the age of a person into account. Figure 18 schematically shows the setup. For both Hyves as well as the survey we have a rating on each entity for each respondent as well as the age of a respondent. For the survey we moreover know what social media a respondent uses. We only use those respondents that indicated using Hyves. Note that we do not alter the age groups defined in Section 3.4.1 and determined upon all respondents, as these are representative for Dutch society.

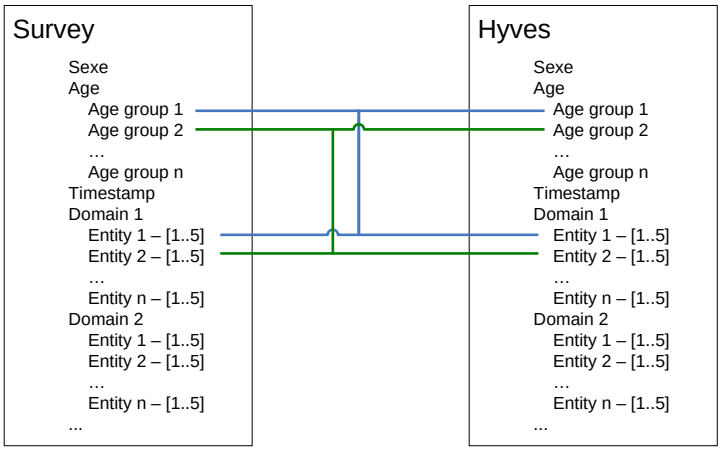


Figure 18: The setup of the segment-based experiments. All age groups are combined with all entities on a per-domain base.

The number of subjective messages extracted from Hyves containing opinions on each of the entities is extremely low for the higher age groups. Figure 30 only shows the fractions of positive, negative and neutral opinions on each entity filled in by respondents of the survey in the age group [18, 29] both because this group is best represented as well as for compactness. Figure 31 shows the same results stemming from our sentiment analysis on Hyves.

Little similarity can be found between Figures 30 and 31 except for the politicians domain. A direct mapping does not match for other domains and even the ordering in which entities within a single domain are rated differs for both sources. For example, the ordering from most positive to least positive for the soccer domain is *FC Barcelona, Manchester United, AC Milan* for the survey whereas it is *FC Barcelona, AC Milan, Manchester United* for Hyves. For the politician domain however, the results do resemble each other quite well. This implies that the way sentiment is constructed differs per domain

and that for the politicians domain, an alignment between the traditional survey and our automated sentiment analysis can be made.

Though not presented, the other age groups are too badly represented in our sentiment analysis to draw any conclusions from. The lack of similarity between our sentiment analysis and the traditional survey may be due to different factors, three of which we list here.

Global Alignment

For our global alignment, we disregard the age and hence regard the complete population rather than an age-specific segment thereof. This is schematically shown for Twitter and Facebook in Figure 19. We also align with Hyves globally, which equals Figure 18 without regarding the age.

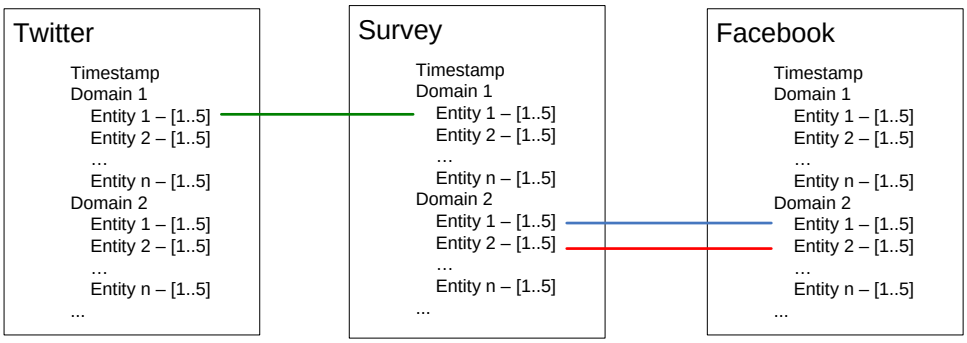


Figure 19: The setup of the global experiments against Twitter and Facebook, age information is discarded. Though not depicted here as it is shown in Figure 18 for the segment-based case, Hyves is compared against globally as well.

The total number of opinions extracted from Twitter, varying from 1 to 1824 per entity, is much higher than for Hyves, varying from 3 to 813 per entity. For Facebook, almost no opinions were extracted. This is most likely due to the fact that very little (publicly available) Dutch messages are available on Facebook. Due to the lack of data, creating an alignment from Facebook is meaningless and hence we only regard Twitter and Hyves globally.

The global level results for Twitter and Hyves differ very little for both the survey as well as the sentiment analysis case. We therefore only show global results for Hyves for conciseness in Figures 32 and 33 for the survey and the sentiment analysis respectively. For our sentiment analysis on Hyves, there is quite some difference with respect to the [18, 29] group, implying that the under-represented age groups (all other than [18, 29]) combined together do influence the results. For the survey, we also see quite some difference with respect to the [18, 29] age group.

When we regard a direct mapping, we see very little similarity, not even with the politicians domain. When we look at the positiveness ranking however, the soccer domain shows some resemblance between our sentiment analysis and the traditional survey. From the latter we cannot draw any conclusion as we cannot clearly identify why this is since we do not have a proper segment-based similarity. For the former however, we can conclude that either the politician domain can only be aligned for certain age groups that are well represented in our sentiment analysis.

Biased Alignment

The segment-based and global alignments make some unrealistic assumptions and use a mapping that is very unlikely to be correct. To make a better mapping, we use the survey's results on one domain to define a mapping for the other domains. We do this as follows.

As we still align each social medium separately, we only regard respondents using the social medium of interest. For one domain we collect all results on all entities. As we use all results, we do not perform any segmentation. We do this because more data will allow us to more easily find boundaries between negative, neutral and positive sentiment for our mapping. For each entity in this domain we define its fraction of positive sentiment as the number of times a rating of 1 or 2 was given divided by the total number of ratings. Negative sentiment is defined similarly, using 4 and 5. Neutral is defined as a rating of 3.

We average the fraction of positive sentiment p, negative sentiment n and neutral sentiment o across all entities in the domain. We now use our sentiment analysis' results on the same domain. All ratings of our sentiment analysis are sorted and the p highest scoring fraction of ratings is defined as positive ratings. We do the same for the n lowest scoring ratings, defining negative ratings. The remaining fraction of ratings o is neutral. The boundary of each class is now used, computed by the mean of the lowest positive rating and the highest neutral rating and the mean of the lowest neutral rating and the highest negative rating. This yields two boundaries, b_{neg} between the negative and neutral class and b_{pos} between the neutral and positive class. As mentioned in Section 2.4.2, we can use these two boundaries and say that any score lower than b_{neg} is negative and any score higher than b_{pos} is positive rather than using 0 as decision boundary.

We use the soccer domain to define our mapping upon. There is no real rationale behind using this domain, we simply pick one. Applying the above mentioned procedure we find $p = 0.620$, $n = 0.325$ and $o = 0.055$ which implies $b_{neg} = -0.567$ and $b_{pos} = 0$ for Hyves. This situation is schematically shown in Figure 20. For Twitter and Facebook we find that $b_{neg} = 0$ and $b_{pos} = 0$ which would yield the exact same setting as the regular global alignment presented in Section 3.4.3. We hence only regard using Hyves.

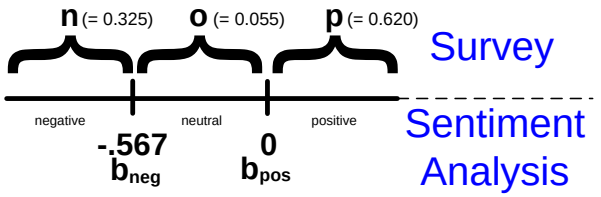


Figure 20: A schematic view on the concrete mapping in case of Hyves. The fraction n, o and p are obtained from analyzing the survey. The values for b_{neg} and b_{pos} are obtained by using the n^{th}, o^{th} and p^{th} fractions of the sentiment analysis results.

For the survey, the sentiment does not change from Figure 32 as we only descretize the scale of the sentiment analysis. For the sentiment analysis we get the results shown in Figure 34. As we trained our mapping on the soccer domain, this domain is not shown. It is easy to see that even though we use a different mapping, the results of Figure 34 closely resemble the results of Figure 33 where we used a decision boundary of 0. Since for our global alignment we concluded that there is very little similarity, this conclusion also holds in this biased setting.

3.4.4 Summary

We tried to make an alignment between our automated sentiment analysis and a traditional survey specifically targeting opinionated marketing research, achieving **Goal 4**. We identified problems that naturally arise when such an alignment is attempted to be made. We showed that sentiment expressed on social media inherently differs from sentiment expressed in traditional surveys. We also showed that sentiment analysis' results not segmented into age groups are unreliable as the distribution of age segment of social media differs drastically from a representative sample of Dutch society. We did not solve or elaborately study the problems we identified as this would be a study on its own which moreover would have a sociological nature rather than a computer science nature.

We used data in which sentiment is expressed in natural language for both the social media as well as the traditional survey. This data we manually labeled and assumed to be ground truth. On this data we assumed the existence of mapping between the two different datasets. This assumption allowed us to compare scores for metrics on the two datasets. We first investigated the distributions of sentiment in both datasets. We found that the sentiment distributions hardly resemble each other but as we assumed a mapping to exist we took this as a reference point. We next performed our sentiment analysis on the social media data and found a weak similarity in the sentiment distributions with the original ground truth dataset, our reference point. Using the mean squared errors as a metric to more concretely define the difference between two distributions showed that our sentiment analysis almost doubles the error with respect to the reference point. We concluded that our sentiment analysis not yet strong enough to establish the mapping we assumed to exist.

We next investigated what the effect is if we deliberately introduce errors into our social media dataset. We mentioned that as the error rate increases, both the negative as well as the positive classes become better represented at the cost of the objective class. Remarkably enough, the mean squared errors decrease as our error rates increase. This could indicate that the mean squared error is perhaps not a suitable comparison metric in this setting. Again we note how a more elaborate sociologically-oriented study on this topic may give better insights and results.

We also analyzed the difference in the two datasets by using accuracy as a comparison metric. We ran our sentiment analysis on both data sources and compared the resulting accuracies to determine whether the sources are comparable. We found that due to the similarity of accuracies, the data is likely similar. Note that this similarity only holds when we use the open field responses of our traditional survey, not when we use the score-based ratings.

In addition to looking only at the sentiment expressed in natural language, we also investigated concrete mappings when using the score-based ratings for our traditional survey. We identified additional problems that arise in this setting. We performed a segment-based alignment in an attempt to overcome the drastic difference in segmental distributions. The only similarity was within the politicians domain, the other domains showed no similarity at all. For the global experiments we saw how the under-represented age groups distorted even the alignment of the politicians domain. In an attempt to create a more realistic alignment we performed a biased experiment where we used the survey data to train a mapping from the survey's opinion scale to the social media's survey scale. Even when we used such a mapping, very little similarity was found which is most likely explained by the inherent difference between social media users and survey respondents.

We already stated in our natural language alignment experiments that it is not likely that a mapping between the score-based ratings of the survey and the natural language of the survey exists. The lack of similarity between the results of our sentiment analysis on natural language and results of the score-based ratings of our traditional survey makes this even more likely. Note though that the lack of similarity we found is not strong evidence for the absence of a mapping between the score-based ratings and the open field responses of our survey as there may be a clear mapping but we just have not found it. This however is unlikely as when performing we manually labeled a large amount of open field responses and found that the sentiment expressed in these open fields disagrees with the rating given approximately equally often as it agrees.

3.5 Summary of Experiments

We performed nine different kinds of experiments divided into two categories: *comparative* experiments and *traditional survey* experiments. In the former type of experiments, we compared our four-step approach on different levels of granularity with other approaches. For the latter type of experiments, we attempted to align automated sentiment analysis results onto traditional survey results at different levels of granularity and using different types of data for our survey.

The most important experiments we have performed are experiments in which we compared each single step, except for the POS-tagging step, of our four-step approach against other competitor approaches and baselines taking accuracy, precision and recall into account. The experiments favored our approaches for each step. The accuracy of our LIGA algorithm for language identification shows an improvement of 4.4% over the competitor approach. The accuracy of our AdaBoost solution to subjectivity detection shows an accuracy 2.1% higher than the most accurate competitor approach we experimented with and an improvement of 16.3% over the baseline approach that showed the best accuracies in our experiments. For polarity detection, our RBEM algorithm outperforms competitor approaches by at least 0.1% when we do not take into account that our model contains too little labeled information. When we do take this into account, the increase in accuracy is at least 11.6%. The improvement over the best performing baseline we experimented with is 4.3% when not taking insufficient labeling into account and 15.8% when we do take the insufficiency of our labeling into account.

The experiments we conducted at the level of the complete process showed that each step present in

our four-step process is beneficial to more accurately solving the sentiment analysis problem at hand. When the complete four-step process is applied, an accuracy of 69.2% was achieved on our test set. This accuracy is in line with what related work typically reports for Naive Bayes or SVM approaches [Pak and Paroubek, 2010, Go et al., 2009, Pang et al., 2002, Esuli and Sebastiani, 2006]. Note though that related work typically only regards sentiment analysis on a single language whereas we perform multilingual sentiment analysis. Moreover, related work often only regards a two way classification, thus actually performing either of subjectivity detection or polarity detection but not both when reporting accuracies. Leaving any of the steps out resulted in worse accuracies where leaving out steps occurring later in the process has more effect on the accuracy. This is not strange as we apply more specialized approaches after each step.

We explicitly quantified the error propagation made at each step. Our experiments show that errors made in language identification have a more severe effect on subsequent steps. This again is because we apply more specialized approaches after each step. Making errors in early steps will most likely only cause more errors in later steps. Our experiments also showed that the use of our four-step process is justified as long as the error rate of any step is below 20%, thus still outperforming competitor approaches solving sentiment analysis in one step.

For a secondary line of experiments, we constructed and conducted a traditional online survey in which Dutch citizens were asked to give their opinion on a set of preselected entities. We performed automated sentiment analysis on the same entities as those listed in the survey. We then tried to make a mapping between the results of the traditional survey and the results of the sentiment analysis process. We identified problems arising with defining such a mapping as the results stem from data that is inherently different in nature. These experiments were an additional investigation aimed to lead to additional validation of our results and make a link to traditional surveying. The main purpose was to set a solid ground for further research in this area.

For these secondary experiments, we regarded samples of the open field responses of a traditional survey as a ground truth by manually labeling them to be either positive, negative or objective. We assumed a mapping between the two data sets exists even though we do not know it explicitly. With the assumption that such a mapping exists we defined a reference point of sentiment distributions of the two different datasets. We then ran our sentiment analysis on the social media data and showed weak signs of similarity, thus indicating that our sentiment analysis is not yet strong enough to establish the mapping we assumed to exist. We also investigated what happens if we deliberately introduce errors and found that our sentiment analysis' errors are not uniform but typically consist of subjective messages being misclassified as objective. This is likely due to the little patterns present in our RBEM algorithm. Our investigation in the similarity of the two data sources we performed our sentiment analysis process on the two datasets and achieved comparable accuracies, indicating that the sources are comparable as well.

The problems we identified for our secondary experiments showed that it is not straightforward to create a mapping either at the global or at a segment-based level when we use score-based ratings for our traditional survey against natural language for our sentiment analysis. Even biasing our experiments by training the mapping on the survey data of entities within one domain did not yield a clear mapping on other entities. The difference in nature between traditional surveying using score-based ratings and automated sentiment analysis, supported by the dissimilarity shown when mapping the two onto each other, motivates using the two in parallel to complement each other rather than preferring one over the other. Automated sentiment analysis should be used to gain insights into the voluntarily provided sentiment present on social media whereas traditional surveying can explicitly extract sentiment perhaps not found on social media by asking survey respondents for it.

4 Conclusions

In this work we have studied the problem of sentiment analysis on multilingual short texts typically present in social media. We presented a four-step approach to solve this problem and presented an extensive experimental study supporting our approach and comparing our sentiment analysis with traditional surveying.

4.1 Main Contributions

Previous work in the area of sentiment analysis traditionally focuses on one language only, usually English as the resources for English are best available. Even though many of these works can be extended to support multiple languages, this is not a trivial task and typically not included in the studies themselves. We presented a multilingual solution to sentiment analysis and demonstrate its applicability in our experiments by taking both English and Dutch into account.

For each of the four steps comprising our solution, we presented an approach to solve it. For part of speech tagging we used an existing solution. For language identification we proposed a new graph-based approach called LIGA. For subjectivity detection we used an existing technique, boosting in the sense of AdaBoost ensemble learning, and applied it to subjectivity detection. For polarity detection we borrowed ideas from previous work to create the rule-based emission model (RBEM) algorithm. This newly presented approach uses eight heuristic rules to efficiently solve the polarity detection problem.

Both of the newly introduced algorithms, LIGA as well as the RBEM algorithm, provide a solid foundation that is easily extended. For LIGA one could incorporate capturing more grammar by enriching the graph. For the RBEM algorithm, adding additional that further increase its accuracy is straightforward when the relation with the other rules is analyzed.

In our experimental study we showed how each of our chosen solutions for each step outperform other competitor approaches and achieve significant performance increase over baselines. We showed the importance of each step, justifying our four-step approach rather than a more generic approach comprising fewer steps. We explicitly quantified what effects it has to get each step as accurate as possible by analyzing the propagation of errors. Not only did we investigate regular error propagation but we also quantified the propagation of composite errors consisting of errors made in more than just one step.

We attempted to align our sentiment analysis' results with those of a traditional survey. We identified inherent problems that arise with such an alignment and propose techniques to overcome these problems whenever possible. We showed how sentiment analysis cannot replace traditional surveying because it is not yet accurate enough. We also identified the dissimilarity in sentiment expressed in natural language and sentiment expressed in a scoring-based fashion.

4.2 Practical Implications

We show that using automated sentiment analysis cannot replace traditional surveying. Part of this result may be due to differences naturally present in the two worlds; in one world a person pro-actively shares an opinion whereas in the other world a person is actively asked for an opinion. Instead of replacing traditional surveying, automated sentiment analysis should rather be used to complement traditional surveying and monitor the entirely different world of social media.

Our experiments align our sentiment analysis results with traditional survey results. For Hyves, we make a segment-based analysis taking different age groups into account. Results such as those shown in Figure 21 can also be refined to different segments, thus allowing for more fine-grained analysis.

Another (very simple) view created using our presented work that allows for more fine-grained analysis is shown in Figure 22. Search terms can be entered and selections based upon profile attributes such

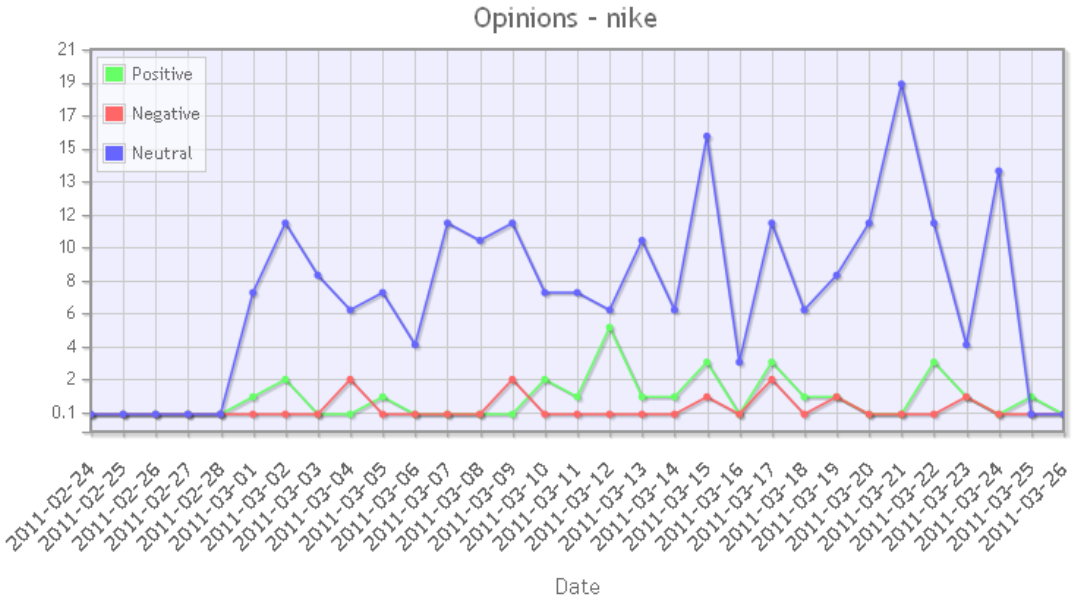


Figure 21: A view created using our sentiment analysis on the crawled data used in our experiments, allowing for easy automated trend monitoring. We searched for opinions on *Nike*.

as *gender* can be made, thus allowing for segment-based analysis. As an example consider querying for sentiment on a term for all Dutch males against all Dutch females. Comparing the results of these queries gives insights in the difference in opinion between males and females, being different segments, in the Netherlands.

Figure 22: A view created using our sentiment analysis allowing for segment-based analysis.

Crawling for the right terms is essential in getting the right data. Image monitoring can be accomplished by crawling for terms related to the brand or company of which the image is to be monitored. Moreover, crawling for competitors' terms as well allows for quick benchmarking. Another application can be to monitor the effect of publicity campaigns of both the brand or company of interest as well as that of

competitors'.

In Appendix C we present five examples in which sentiment analysis can be used. With each example we list the strengths and weaknesses of using sentiment analysis rather than traditional surveys. We identify general motivations of using sentiment analysis over traditional surveying that apply to all examples. These motivations are the continuity sentiment analysis provides, the little human interaction required, the little time required to obtain results and the ease of obtaining much larger amounts of data.[13]

We present an example on monitoring of sentiment which directly demonstrates the strength of the continuity of sentiment analysis. We argue how traditional surveying cannot be performed with such continuity. We also mention how segmenting is easy and comes at no cost with sentiment analysis but requires a costly maintenance of a panel for traditional surveying. On the other hand, with a traditional survey we can more directly ask respondents what exactly their opinion is on whereas with sentiment analysis we have to rely on what is voluntarily given to us.

We show how benchmarks can be created through sentiment analysis to quickly gain insights into market stances. The benefit of using sentiment analysis in this setting is that we can continuously monitor market stance rather than periodically, allowing for a more agile response to shifts. Additional segmentation of benchmarks can give deeper insights into where the strengths of our own brand or product are and where there is a need for improvement.

We present an example on public polling using sentiment analysis, inspired by the work of [O'Connor et al., 2010]. The example describes how sentiment analysis can be used to create public polls on an upcoming election. As a lot of activity in the political work emerges with a nearing election, polls change rapidly and for politicians it is important to identify shifts in the polls as quickly as possible. Our sentiment analysis allows for a continuous up-to-date view on the polls. To create representative polls however, we require to have enough responses for each segment to construct a representative sample. As lower age groups are generally over-represented on social media whereas higher age groups are typically under-represented, we may have difficulties in obtaining a representative sample when using sentiment analysis. This is a problem typically not present when using traditional surveys although then higher age groups typically are better represented.

As an extension on the polling example, we show how sentiment forecasting can be used to predict future sentiment. We show how additional techniques such as regression can give insights into how sentiment will develop in a near future. We note that as additional techniques are required to perform such forecasting that typically do not depend on how the data is gathered, it makes little difference whether we use sentiment analysis or traditional surveys for the forecasting itself.

In our last example we identify a new application field where traditional surveying cannot be used but sentiment analysis can be applied perfectly fine. A typical trend is that people more and more share experiences and problems with products or companies on social media rather than writing a letter, calling a service desk or even sending an e-mail. Nowadays web care is thus becoming an increasingly important aspect of customer care. We show how automated sentiment analysis can be used to support in performing such web care. In an extreme case, such adequate web care can prevent the situations such as that of T-Mobile and comedian van 't Hek, discussed in Section 1.1.

4.3 Future Work

The RBEM algorithm we present uses eight simple heuristics. These eight rules show to work effectively in our experiments but whether these eight rules are the best eight rules to use is not clear. Additional rules may increase the performance. Additionally one can borrow ideas from the Brill tagger [Brill, 1992] where the best possible combination of a set of predefined rules is determined in an automated way.

[13] For our traditional survey experiments, we ran a survey for three days and crawled social media for three days. We obtained a total amount of 1429 responses to the survey whereas we obtained little over 400 thousand relevant – that is, messages on the entities we were looking for – messages.

The models we used for the RBEM algorithm consisted of a relatively small amount of data as labeling is very labor-intense. We have shown that more labeling increases the accuracy. It would thus be beneficial to extend our models of the RBEM algorithm.

For our experimental study we constructed different datasets. The training set was constructed by querying Twitter with smilies and scraping news accounts. This yields messages with noisy labels rather than accurate labels. Moreover, this training data is biased as it only contains messages in which smilies are originally present. Constructing more accurate and more representative training data is needed to increase the realism of our study. Additionally, data can be collected for each social media separately, allowing for more specific, social medium-tailored, models. In order to strengthen the conclusions made in this paper, the size of the validation and ground truth sets should be bigger.

In our traditional survey experiments we only regard sentiment on given entities. An entity as we proposed it is an object one is interested in. It is more realistic however that we are not just interested in such an object, but additionally in features of that object. Consider for example having a camera as object of which we would like to perform sentiment analysis on one of its parts, it's shutter for example. Such fine-grained analysis is not trivially possible with the techniques we propose. Extending our proposed techniques could answer this problem however.

We show that automated sentiment analysis cannot replace traditional surveying and identify that an alignment between sentiment analysis results and traditional survey results is hard to make. We identify problems that arise when making such an alignment. Our discussion on these problems is not as elaborate as it could be. A detailed study into this topic should be performed by academics from sociological sciences rather than computer science. The results of such a study may give additional insights in the difficulties and potential solutions of the alignment between the traditional survey and our sentiment analysis.

References

[Ahmed et al., 2004] Ahmed, B., Cha, S., and Tappert, C. (2004). Language identification from text using n-gram based cumulative frequency addition. In *Proc. CSIS'04*.

[Brill, 1992] Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*.

[Butterworth et al., 2004] Butterworth, R., Simovici, D. A., Santos, G. S., and Ohno-machado, L. (2004). A greedy algorithm for supervised discretization. In *Journal of Biomedical Informatics - Special issue: Biomedical machine learning*.

[Cavnar and Trenkle, 1994] Cavnar, W. and Trenkle, J. (1994). N-gram-based text categorization. In *Proc. 3rd Symp. on Document Analysis and Information Retrieval (SDAIR-94)*.

[Cowie et al., 1999] Cowie, J., Ludovic, Y., and Zacharski, R. (1999). Language recognition for mono- and multilingual documents. In *Proc. of the Vextal Conference*.

[Cutting et al., 1992] Cutting, D., Kupiec, J. M., Pedersen, J., and Sibun, P. (1992). A practical part-of-speech tagger. In *In Proceedings of the Third Conference on Applied Natural Language Processing*.

[Dougherty et al., 1995] Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*.

[Dunning, 1994] Dunning, T. (1994). Statistical identification of language. In *TR-MCCS-94-273, New Mexico State Univ.*

[Esuli et al., 2010] Esuli, A., Baccianella, S., and Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation LREC'10*.

[Esuli and Sebastiani, 2006] Esuli, A. and Sebastiani, F. (2006). Determining term subjectivity and term orientation for opinion mining. In *Proceedings EACL-06, the 11rd Conference of the European Chapter of the Association for Computational Linguistics*.

[Freund and Schapire, 1995] Freund, Y. and Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting.

[Go et al., 2009] Go, A., Huang, L., and Bhayani, R. (2009). Twitter sentiment analysis.

[Goldwater and Griffiths, 2007] Goldwater, S. and Griffiths, T. L. (2007). A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

[Harper and Teahan, 2001] Harper, D. and Teahan, W. (2001). Using compression-based language models for text categorization. In *Proc. Workshop on Language Modeling and Information Retrieval*.

[Hatzivassiloglou and McKeown, 1997] Hatzivassiloglou, V. and McKeown, K. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the ACL*, pages 174–181.

[Joachims, 1998] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning (ECML)*.

[Joachims, 1999] Joachims, T. (1999). *Making large-scale support vector machine learning practical. Advances in Kernel Methods - Support Vector Learning*. MIT Press Cambridge. B. Schlkopf and C. Burges and A. Smola.

[Kupiec, 1992] Kupiec, J. M. (1992). Robust part-of-speech tagging using a hidden markov model. In *Computer Speech and Language, Vol. 6*.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*.

[Martino and Paulsen, 1996] Martino, M. and Paulsen, R. (1996). Natural language determination using partial words. In *US Pat. 6216102 B1*.

[O'Connor et al., 2010] O'Connor, B., Balasubramanyan, R., Routledge, B. R., and Smith, N. A. (2010). From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.

[Pak and Paroubek, 2010] Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC 2010*.

[Pang and Lee, 2004] Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

[Pang and Lee, 2008] Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. In *Foundations and Trends in Information Retrieval 2(1-2)*.

[Pang et al., 2002] Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP02*.

[Potena and Diamantini, 2010] Potena, D. and Diamantini, C. (2010). Mining opinions on the basis of their affectivity. In *2010 International Symposium on Collaborative Technologies and Systems (CTS)*, pages 245–254.

[Prager, 1999] Prager, J. (1999). Linguini: Language identification for multilingual documents. In *Proc. 32nd Hawaii Int. Conf. on System Sciences*.

[Quinland, 1983] Quinland, J. (1983). Learning efficient classification procedures and their application to chess end games. In *Machine Learning: An Artificial Intelligence Approach*. Edited by R. S. Michalski and J. G. Carbonell and T. M. Mitchell.

[Quirk et al., 1985] Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. (1985). *A Comprehensive Grammar of the English Language*. Longman, London.

[Read, 2005] Read, J. (2005). Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *ACLstudent '05 Proceedings of the ACL Student Research Workshop*.

[Riloff et al., 2003] Riloff, E., Wiebe, J., and Wilson, T. (2003). Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 25–32.

[Schapire and Singer, 2000] Schapire, R. E. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. In *Machine Learning*.

[Schmid, 1994] Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*.

[Sindhwani and Melville, 2008] Sindhwani, V. and Melville, P. (2008). Document-word co-regularization for semi-supervised sentiment analysis. In *Eighth IEEE International Conference on Data Mining*.

[Tromp and Pechenizkiy, 2011] Tromp, E. and Pechenizkiy, M. (2011). Graph-based n-gram language identification on short texts. In *Proceedings of the 20th Machine Learning conference of Belgium and The Netherlands*.

[Wiebe and Micalcea, 2006] Wiebe, J. and Micalcea, R. (2006). Word sense and subjectivity. In *Proceedings of ACL06*.

[Wiebe et al., 2005] Wiebe, J., Wilson, T., and Cardie, C. (2005). Annotating expressions of opinions and emotions in language. language resources and evaluation. In *Language Resources and Evaluation (formerly Computers and the Humanities)*.

[Wilson et al., 2005] Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

A Framework

A.1 Introduction

In this appendix we present the framework constructed with and implementing the sentiment analysis theoretically presented in this work. We first start by giving a high-level overview, describing the relations between the different components, in Appendix A.2. We give instructions on installation and maintenance of our framework in Appendices A.3 and A.4. Thereafter we describe each of the components separately and for each component point out its purpose, goal and realization thereof.

A.2 High-level overview

Figure 23 shows a high level overview of all components and their relations. We define three different processes: *Data collection*, *Data processing* and *Data analysis*. The *Install* class is a single webpage that sets options used with the data collection. This must be done before any data collection is started. The *Initiator* class is a single command line script that invokes all crawling and scraping activities and should be called once to start all data collection in parallel. The *Miner* class is a single command line script similar to the *Initiator* webpage and should be called once to start all data processing in parallel. Also note the *constants.php* file that needs to be configured before first use.

The *Data collection* process is concerned with obtaining the data. This entails communicating with the social media to obtain the right data and storing the data in a uniform format. The process consists of first crawling the social media and later scraping it. Crawling is mainly concerned with identifying sources (within the social medium) from which data can be extracted. Scraping is then concerned with actually extracting the data. The following components belong to this process.

- Crawler
  - twitterCrawler
  - hyvesCrawler
  - facebookCrawler
- Scraper
  - twitterScraper
  - hyvesScraper
  - facebookScraper
- Twitter interface
- Hyves interface
- Facebook interface
- OAuth

The *Data processing* process is concerned with processing the collected data. Processing the data begins with identifying the language of a single text. Next, a language-specific POS-tagger tags all entities within a text. Using this information, a subjectivity detector will determine whether a text is subjective or objective and. Finally, the polarity of all subjective texts is determined by a polarity detector. Though these processes run in parallel, for a single text they should be executed successively. The following components belong to this process.

- LanguageIdentifier

  - LIGA
- POS Tagger
  - TreeTagger
- Subjectivity detector
  - AdaBoost
- Polarity detector
  - RBEM

The *Data analysis* process is concerned with interpreting the results from the *Data processing* process. This is done at the presentation level as different presentations of the resulting data imply different interpretations. The *Web* package deals with this.

All three processes deal with data stored in a database warehouse. The data warehouse consists of four databases: *Crawler*, *Data*, *Data* and *Mining*. The *Crawler* database stores data resulting from the crawling process which is part of the *Data collection* process. The *Data* database holds data resulting from the scraping process, part of of the *Data collection* process. The *Mining* database stores information resulting from the *Data processing* process. The *Front* database stores information on the *Data analysis* process.

The *Data collection* and *Data processing* processes should be performed indefinitely whereas the *Data analysis* should be triggered. The *Data collection* and *Data processing* processes do not communicate with the database directly but make use of four different interfaces: *work*, *store*, *portfolio* and *mining*. This is to allow for a problem-less distributional approach. The *work* interface is used to fetch work for a specific process, it is a read-only interface. The *store* interface is to store data resulting from a specific process, it is a write-only interface. The *front* interface is used when dealing with settings regarding the *Data analysis*. Finally, the *mining* interface is used by the *Data processing* process.

A.3 Installation

To get the software package up and running, it first needs to be configured properly. To this end the *constants.php* file needs to modified after which the *Install* script should be ran. Note that the only publicly available folders should be the *web* and *Model* folder.

The *constants.php* file contains definitions that need to be set correctly. We describe each of these definitions.

ROOT_LOCATION This is the absolute (file) path to the root location (in which the *constants.php* file is located) of the system.

CRAWLER_DB This is the name of the crawler database.

DATA_DB This is the name of the data database.

FRONT_DB This is the name of the front database.

MINING_DB This is the name of the mining database.

MYSQL_HOSTNAME The location (hostname) of the MySQL database.

MYSQL_USERNAME The username to log into the MySQL database.

MYSQL_PASSWORD The password to log into the MySQL database.

MODEL_LOCATION The (possibly external) absolute path to the model.

Figure 23: A high level overview of the software framework constructed in this project.

WEB_LOCATION The location of the front-end web interface, this value can be altered but usually should not have to be altered.

PHP_PATH The (shell) command to run command line PHP.

LOG_PATH The absolute path to the logging directory, usually should not be changed.

Once the *constants.php* file is configured correctly, the installation can commence. This is done by running the *index.php* file of the *Install* module. This installation contains a step-by-step guide to set up the social media accounts. Note that *Linkedin* is currently not supported and hence configuration for this social medium is not obligated. Note moreover that the credentials entered and validated in this installation procedure are cached in the *Cache* folder and hence it is wise to use dummy accounts to this end. After installation, the *Install* folder can be removed, which is highly encouraged.

The software is designed to be used in a distributed fashion. There are three main components that can be separated from each for obvious efficiency reasons other but may just as well be located on one single server. The first is the most important and contains the data model. One (farm of) server(s) can act as data model. In this case, the *Model* folder as well as the database itself, as described in Appendix A.9, should be present on the server.

The second component is that of the scrapers and crawlers. The *initiator.php* and *miner.php* initiate all crawling and scraping actions. A single server can perform a single action from either the initiator or miner file or can perform multiple such actions. At the extremes, one server can perform all actions or a single server can be used for each action separately. The actions a server performs are defined in the *$actions* arrays in both the *initiator.php* file as well as the *miner.php* file. The crawling and scraping servers not necessarily need the *Model* folder and the database accessible but do need the *initiator.php* file, the *miner.php* file, the *Crawler* folder, the *Scraping* folder and all social media classes.

The third component entails all web interfaces located in the *web* folder. If the web and model components are to be explicitly separated, the *Model* folder should contain interfaces for all web interfaces to communicate with. This is currently not the case for the few web interfaces that are present.

A.4 Maintenance

As with any software package, this package needs to be maintained. The most important maintenance is in the models of the different supervised learning tasks used; language identification, POS-tagging, subjectivity detection and polarity detection. Another aspect that may need modification over time is the communication with the social media as the APIs of these may change over time. We first describe how the learned models can be updated or replaced in Appendix A.4.1 and next continue with how changes in the APIs of social media should be dealt with in Appendix A.4.2.

A.4.1 Model Updating

Currently there are differing number of models present with each of the four steps our sentiment analysis comprises. For language identification, we have support for six different languages; *English, Dutch, German, Italian, French* and *Spanish*. For POS-tagging we have support for all languages present on the TreeTagger website[14], currently being *English, Dutch, German, Italian, French, Spanish* and *Portuguese*. For the subjectivity as well as the polarity detection however, we only have support for two languages; *Dutch* and *English*.

To update the models of the above mentioned languages or to add new language models, different steps need to be taken depending on which of the four steps the model is to be updated or expanded. Whenever applicable, our software supports in doing this. We describe how to do this for each step separately. The

[14] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

actual models for language identification, subjectivity and polarity detections can be physically found in the *Cache* folder whereas the models for the POS-tagger can be found in the *POS/TreeTagger/bin*.

Language Identification

The language identification is implemented in the *LanguageDetector/LanguageIdentifier.php* class. This is a static class that facilities training models. If a language is to be added, or a model is to be extended, the current model should first be loaded by calling the *loadModel()* function which loads the entire model into memory. Next, the *addDocument($document, $language)* function can be called for each message that is to be added to the model. The *$document* argument should be a string containing the message to add whereas the *$language* argument should be the code of the language (eg. *nl_NL* for Dutch or *en_UK* for English). Once all messages have been added to the model, it should be stored by calling the *storeModel()* function. If the current model is to be overwritten, the *loadModel()* function should not be called.

POS-tagging

As the POS-tagger we use is not developed by us, we refer to the TreeTagger website for more information on how to train models for this algorithm. Note that numerous pre-learned models can be downloaded from the website.

Subjectivity Detection

The subjectivity detection is performed using AdaBoost. This algorithm is implemented in the file *OpinionMiner/AdaBoost.php*. This file not only contains a class implementing the AdaBoost algorithm but also a function to train a model; *Subjectivity($lang)*. Though it is not required to use this function instead of writing your own script to train the AdaBoost model, it shows a good example. As the AdaBoost algorithm learns a language-specific model, it requires a language code passed via the *$lang* argument.

The function to train an AdaBoost model is implemented in the *train($data, $labels, $numLearners)* function of the AdaBoost class. This function requires all the data to be put into the model in the $data argument as an array containing all instances. An instance in turn consists of an array whose keys are features and whose values do not matter but are typically the constant 1. The $labels argument is an array that contains the labels of all instances in $data in the same order. The $numLearners argument defines the number of weak learners to train.

Polarity Detection

The polarity detection algorithm uses patterns as a model. The algorithm is implemented in the *OpinionMiner/EmissionMiner.php* file. A model for a language *$lang* can be expanded by first loading it using the *loadModel($lang)* function. Patterns can then be added using the *addPattern($pattern, $type)* function where *$pattern* an associative array of the pattern where the keys *token* and *tag* indicate which is the token and which is the POS-tag respectively and the *$type* argument is the type of pattern. A model can be stored again using the *storeModel($lang)* function.

A small and simple helping tool to add patterns to a model is implemented as a web interface in the *web/emissionTrainer.php* file which can be accessed as a website to add patterns in through a two-step process. As a GET parameter to this web interface the *lang* parameter should hold the code of the language for which patterns are to be added. An example request for adding English patterns is *web/emissionTrainer.php?lang=en_UK*. The web interface highlights known positive patterns as green and known negative patterns as red.

Below each token a checkbox is shown. To add a pattern consisting of tokens $t_1, .., t_n$, all checkboxes shown directly under each t_i should be checked. Next, the dropdown bow located to the right of the message should be used to specify the type of the pattern. When the *Group* button is then clicked, the pattern will be created for inspection in the next step. Once all messages shown have been processed and all present patterns have been grouped, the *Incorporate check into models* button should be clicked. This presents the second step of adding patterns. For all patterns grouped in step one, the type can be inspected and possibly corrected and wildcards can be set. Once all patterns have been inspected and correctly set, the *Add to model* button should be clicked which effectively adds the patterns to the model.

A.4.2 Social Media APIs

Our software currently supports three social media; *Twitter, Facebook* and *Hyves*. Each of these social media have their own APIs to communicate with. Interfaces performing this communication are implemented for each social medium separately in the folder *%SM%/%SM%.php* where *%SM%* should be replaced with the name of the social medium. All functions, including both crawling and scraping, that are used by our software are implemented in these files. Any changes to the API of a social medium or any additional API support should hence be applied in these files. For Twitter, the API documentation can be found at http://dev.twitter.com/. For Facebook, it can be found at http://developers.facebook.com/. For Hyves, the API documentation is located at http://www.hyves-developers.nl/documentation/data-api/home.

A.5 Data Model

A fundamental aspect of any information system is the data model. In Figure 23, this is represented by the large database on the right. Any communication with the data model is done through its interfaces: *work*, *store*, *portfolio* and *mining*. These interfaces directly communicate with the database whereas entities using these interfaces can be distributed over multiple other resources, allowing for parallelism and scalability. The interfaces read and write to four databases; *crawler*, *data*, *front* and *mining*. Communication through the interfaces is done through certain protocols. For more details on these protocols, please see Appendix A.10. The structure of the databases is given in Appendix A.9.

We allow parallelism but want to avoid duplicate work. Any item that is assigned to a process by one of the interfaces gets a unique code. This code is stored in the *InProcess* column. The *StartProcess* column stores the timestamp at which the item was assigned. We can now check for entries of which either the *InProcess* column is the empty string or for which the *StartProcess*' value is greater than some threshold (e.g. 10 minutes) ago. This check allows us to exclude any redundant work being done when parallelism comes into play.

We first describe each database's purpose and structure and next describe the interfaces' purposes and structures.

A.5.1 Crawler database

The crawler database purely stores raw data extracted from the social media. The database's structure is shown in Appendix A.9.1.

The tables facebook, hyves and twitter have the exact same structure. These tables also store the exact same information but for different social media, indicated by their respective table names. IDs of entities to further inspect are stored along with a state and a last action timestamp indicating the last update time. Generally taken, state 0 indicates the entity has been found but not processed at all yet. State 1 indicates it has been crawled. State 2 then indicates it has been scraped. State 3 indicates scraping was not possible (eg. due to privacy restrictions).

The remaining table, *facebook_terms*, stores random terms found while crawling the Facebook timeline. A state of 0 means this term has not been used in a search yet whereas a state of 1 means it has.

A.5.2 Data database

The data database stores general scraping information. This includes scraped texts that are ready for data processing as well as profiling information for coupling of scraped texts to profiles. The structure is given in Appendix A.9.2.

The *_texts* suffixed tables contain the texts scraped from the respective social medium. The structure of each of these tables is identical. A unique identifier is attached to each text. A timestamp indicating when the text was placed on the social medium (not the timestamp at which the text was found) is added. The source field indicates the author of the text and is a link to the respective *_profiling* suffixed table, allowing for coupling of texts to profile information. A language field indicates the language the text is in. Upon insertion, this field is empty. The language identification process of the data processing process fills this field.

The *_profiling* suffixed tables store any information that can be publicly retrieved for an individual entity for a specific social medium. The ID field present in these tables is the ID of entity as specified in the social medium. This ID also serves as a relation to the *_texts* suffixed tables.

A.5.3 Front database

The front database is generally used for presentation and feedback related issues. Any data stored in this database is not related with the indefinite processes but rather with triggered processes present in the data analysis process. The structure is given in Appendix A.9.3.

The only table present is the *term_portfolio* table. This table is used by crawlers and scrapers to track certain terms or keywords. These terms cannot be altered by the indefinite processes but only by human interaction. Altering these terms will imply a change in tracking terms for the crawling and scraping processes.

A.5.4 Mining database

The mining database stores all data that is used in the data processing process and is presented in the data analysis process. The structure is given in Appendix A.9.4.

The *pos_tags* table is populated by the POS-tagging process. Entries present in this table are tokenized texts of which every token has an accompanying POS-tag vector. The text field contains the original text where every token is separated by a space. The POSTags field contains all found POS-tags. Every token is given one or more POS-tags, separated by a tab, which are listed on separate lines. The creation field is the timestamp of creation of the text whereas the author field contains the ID of the author. The source field indicates the source of the text, used for coupling texts to profiling information. The language field contains the language of the text whereas the boolean UsedInOM field indicates whether or not this text is used in creating a model for subjectivity or polarity detection.

After subjectivity detection has taken place, three possibilities can occur. It may be the case that no model was found for subjectivity detection. In this case, the *Subjectivity* flag in the *pos_tags* table gets the value 2. When a model is found, a text can either be classified subjective (1) or objective (-1). When the text is *subjective*, it is copied to the *opinions* table. The *Subjectivity* field is then the score of the subjectivity classifier, indicating the strength of the classification. When the text is *Objective*, it is copied to the *facts* table where the *Subjectivity* field again indicates the strength. In case a model was found, the texts is removed from the *pos_tags* table.

When a text is in the *opinions* table, it can be retrieved for polarity detection. The resulting polarity is stored in the *Polarity* field. The strength is stored in the *PolarityScore* field. The input of the algorithm is stored in the *PolarSentence* field for easy access later on. The *Emissions* field contains the emission scores for each token.

A.5.5 Work interface

The work interface is used to request work from the database for use with crawling and scraping. Typical requests to be made to this interface involve getting a bag of user IDs which are yet to be crawled or scraped. Appendix A.10 shows the protocol to be used with this interface. If this interface is to be extended, adding a function name will introduce a call as well since protocol calls are translated to function calls on a one-to-one basis.

A.5.6 Store interface

The store interface is the most extensive interface and is used to store any updated or newly obtained data. This not only includes data collected from crawling or scraping but also updates of labels during any of the data processing processes. A specific use of this interface is to update the state of specific entries, indicating a given process has finished and a new process can commence on the specific entry.

A.5.7 Portfolio interface

The portfolio interface is purely concerned with term portfolios. The plural here is due to the terms kept track of for facebook crawling as described in Appendix A.6.1. Terms from *term portfolio* can be requested or set using this interface.

A.5.8 Mining interface

The mining interface is similar to the work interface but is concerned with all data processing tasks, whereas the work interface is concerned with all data collection tasks. Any mining work, requesting a bag of texts to language identify for example, is requested through this interface.

A.6 Data collection

Data is collected from the social media *Hyves*, *Twitter* and *Facebook*. The data is collected in two phases; *Crawling* and then *Scraping*. Each of the social media can be accessed through an API making use of the *OAuth*[15] protocol for authentication. OAuth makes use of consumer keys and consumer secrets. Using this consumer secret and consumer key, a client can contact the OAuth server for an authentication token that identifies the client. Traditionally an OAuth client authenticates itself every session. An authentication token however has a maximum lifespan after which it expires. API requests can then no longer be made. In order to maintain persistency here, OAuth credentials are cached to be reused later.

Once the OAuth credentials are present, data collection can commence. This is a process that should be performed indefinitely. Traditionally, crawling should be performed before scraping. As we will see however, due to the nature of a social medium, we may revert this ordering. We next describe how we perform crawling and scraping.

All data collection is being initiated by the *initiator.php* script.

A.6.1 Crawling

Crawling is the process of extracting content with the aim to find more content. In our case, this means we want to *explore* the social medium in question. For *Hyves* this means we seed with one single user ID

[15] http://oauth.net/

from which the network is being explored by iteratively requesting the connections of discovered, but not yet processed users. For *Facebook* and *Twitter* however, due to their nature, we crawl users that were found after examining a so-called *timeline*. This timeline is a collection of texts ordered by time, along with their authors. We extract the connections of such authors to further explore the network.

Twitter For Twitter, the crawling and scraping are reversed in order compared to Hyves and Facebook. Crawling Twitter thus means we extract profiling information from users found during the scraping of the timeline. We do so through the Twitter interface which communicates with the Twitter API. Note that the Twitter API only allows 300 requests per hour (per IP, hence a distributed approach would be useful).

To determine what users to crawl, we first make a request to the *work* data interface. This returns a list of user IDs from the *crawler.twitter* table which have to be explored. Information extracted from Twitter during crawling is stored in the *data.twitter_profiling* table through the *store* data interface.

Facebook For Facebook, we crawl the timeline to discover the network. We do this using Facebook's API to search for public terms. We query the API with terms present in our *term portfolio* accompanied with some random terms (taken from the *crawler.facebook_terms* table) found in earlier crawling of Facebook. This way we discover more of the network and we do not restrict ourselves solely to terms from the portfolio.

The terms are requested through the *portfolio* interface which returns both portfolio terms as well as some random terms encountered so far. The texts resulting from querying Facebook's API with the given terms are stored in the *data.facebook_texts* table. Moreover, new terms are added to the *crawler.facebook_terms* table and authors of the found texts are put into the *crawler.facebook* table.

Hyves Crawling Hyves consists of iteratively requesting connections of a previously uncrawled Hyves user. Connections are then stored in the *crawler.hyves* table and can be crawled in a next iteration.

A.6.2 Scraping

The scraping process is generally concerned with extracting information of entities which have been discovered as containing useful information during the crawling process. Scraping such an entity entails requesting the information we are interested in and storing it in the database. The methodology behind it is similar to that of crawling. The social media's APIs are used to obtain the information from. The only difference here is that different methods are used. As mentioned in Appendix A.6.1, for Twitter however, scraping entails exploring the timeline.

Twitter Scraping Twitter is done through so called *streaming APIs*. In contrast to traditional APIs, streaming APIs do not give a single result but maintain a persistent connection and continuously push results into this connection. Traditional HTTP connections are meant to be non-persistent and hence this poses a problem with respect to traditional API requests, which can be done through HTTP(S).

Two streaming API methods are used, the *firehose* and *filter* methods. The former streams a statistically sound portion of all publicly available tweets whereas the latter allows to specify certain *tracks*. These tracks are (collections of) terms that are tracked. All tweets returned by the filter method contain one or more of the specified tracks. This suits our term portfolio well.

Since the streaming nature of this API poses some issues, we make use of *Phirehose*[16]. This is a PHP-built interface to the (streaming) Twitter API. We discriminate two processes performed by Phirehose;

[16] http://code.google.com/p/phirehose/

collection and consumption. The former collects data from the Twitter streaming APIs. The latter consumes and processes this information. These two processes are separated since processing of data during collection would take too many resources and we would fall behind in collecting the data. The collection and consumption are realized through a *GhettoQueue*, a simple disk-based queue.

Facebook Scraping Facebook consists of requesting profiling information about a Facebook entry. A fixed amount of IDs is requested through the work interface. Information on gender and locale are then requested through the Facebook API for each of these IDs. This information is then extracted and passed on to the store interface to store the information. Finally, the state is updated to indicate that the entry has been scraped.

Hyves Scraping Hyves consists of somewhat more work than scraping facebook. Similarly, users are scraped for profiling information such as gender, location and birth date. For Hyves however, we also scrape a user's *WWWs* ('Wie-Wat-Waar's) which are small status updates posted by users. We also scrape *Scraps* which are messages posted by one user to another.

In addition to scraping Scraps and WWWs for each user, we also monitor the public WWW-timeline specifically for terms in our term portfolio. This happens in a fashion similar to what is being done with Twitter but now we do not deal with streaming data but regular API calls.

A.7 Data processing

The most interesting aspect of the system is the data processing phase. This process serves as the translation from raw data into meaningful data that can be analyzed or presented. The data processing phase is divided into four steps: *Language Identification*, *Part of Speech tagging*, *Subjectivity Detection* and finally *Polarity Detection*. Even though these steps can be executed on different entries (texts) in parallel, a single entry has to undergo them in the order given.

All data processing is being initiated by the *miner.php* script.

A.7.1 Language Identification

The most important function of the Language Identifier is the *identify* function, which takes as input a text and outputs a language. Before doing so however, a model should be learned by adding documents whose language is known using the *addDocument* function. The model resulting from this learning can be cached using *storeModel* and loaded using *loadModel*. Additionally, the threshold can be altered using *setThreshold*.

A.7.2 Part of Speech tagging

In order to use TreeTagger, the *POSTagger* class serves as an interface. Simply calling its function *tag* will run the TreeTagger and POS tag a given text. Note that the POS-tagger can only tag messages in languages it has models of.

A.7.3 Subjectivity Detection

A bag of unclassified texts is requested from the *mining* interface. Each text is processed by first loading the *AdaBoost* model of the language the text is in. After the features have been extracted correctly, the text is classified by applying the decision stumps with their respective weights. The resulting class is

either *subjective* or *objective*. An optional parameter is used to obtain the certainty of the label. The text's information is aided with this label and its strength and is then posted to the *store* interface. This interface then either places it in the *facts* table (in case of an objective label) or in the *opinions* table (in case of a subjective label).

A.7.4 Polarity Detection

An unclassified sentence is partitioned in order into (token, POS-tag) pairs and classified according to the RBEM algorithm. The result is a list of positions with their respective emission rates. The sign of the sum of these values indicates the class. If the sum is greater than 0, the sentence is said to be *positive*. If the sum is less than 0, the sentence is said to be *negative*. When the sum is exactly 0, this can either indicate that no pattern could be matched against the model or that the negative emissions rule out the positive emissions. These two can be discriminated by looking at the number of patterns that could be matched.

A.8 Data analysis

The data analysis in fact comprises views on the data present after all data processing has taken place. Sentiment information can be combined with profiling information.

A.9 Databases

A.9.1 Crawler

facebook

```
‘ID‘ varchar(255)
‘State‘ int(10)
‘LastAction‘ int(10)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
PRIMARY KEY (‘ID‘)
```

facebook_terms

```
‘Term‘ varchar(150)
‘State‘ int(1)
‘Created‘ int(10)
KEY ‘Term‘ (‘Term‘)
```

hyves

```
‘ID‘ varchar(50)
‘State‘ int(10)
‘LastAction‘ int(10)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
PRIMARY KEY (‘ID‘)
```

twitter

```
‘ID‘ int(32)
‘State‘ int(10)
‘LastAction‘ int(10)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
PRIMARY KEY (‘ID‘)
```

A.9.2 Data

facebook_profiling

```
‘ID‘ varchar(255)
‘Gender‘ enum(’male’,’female’,’unknown’)
‘Locale‘ varchar(10)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
PRIMARY KEY (‘ID‘)
```

facebook_texts

```
‘ID‘ int(10)
‘Text‘ text
‘Timestamp‘ int(10)
‘Source‘ varchar(255)
‘Language‘ varchar(25)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
PRIMARY KEY (‘ID‘),
UNIQUE KEY ‘Timestamp‘ (‘Timestamp‘,‘Source‘),
KEY ‘Source‘ (‘Source‘)
```

hyves_profiling

```
‘ID‘ varchar(50)
‘Gender‘ enum(’Male’,’Female’,’Unknown’)
‘Country‘ varchar(255)
‘City‘ varchar(255)
‘BirthDay‘ smallint(2)
‘BirthMonth‘ smallint(2)
‘BirthYear‘ smallint(6)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
PRIMARY KEY (‘ID‘)
```

hyves_texts

```
‘ID‘ int(10)
‘Text‘ text
‘Timestamp‘ int(10)
‘Source‘ varchar(50)
```

```
‘Language‘ varchar(25)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
PRIMARY KEY (‘ID‘),
UNIQUE KEY ‘Timestamp‘ (‘Timestamp‘,‘Source‘),
KEY ‘Source‘ (‘Source‘)
```

twitter_profiling

```
‘ID‘ int(10)
‘Language‘ varchar(10)
‘Location‘ varchar(255)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
‘Description‘ text
PRIMARY KEY (‘ID‘)
```

twitter_texts

```
‘ID‘ int(10)
‘Text‘ varchar(255)
‘Timestamp‘ int(10)
‘Source‘ int(32)
‘Language‘ varchar(25)
‘InProcess‘ varchar(6)
‘StartProcess‘ int(11)
PRIMARY KEY (‘ID‘),
UNIQUE KEY ‘Timestamp‘ (‘Timestamp‘,‘Source‘),
KEY ‘Source‘ (‘Source‘)
```

A.9.3 Front

term_portfolio

```
‘ID‘ int(10)
‘Term‘ varchar(255)
PRIMARY KEY (‘ID‘)
```

A.9.4 Mining

facts

```
‘ID‘ int(11)
‘Text‘ text
‘POSTags‘ text
‘Creation‘ int(11)
‘Author‘ varchar(64)
‘Source‘ enum(’hyves’,’twitter’,’facebook’)
‘Language‘ varchar(10)
‘Subjectivity‘ float
‘InProcess‘ varchar(6)
```

```
`StartProcess` int(11)
PRIMARY KEY (`ID`),
UNIQUE KEY `Creation` (`Creation`,`Author`,`Source`)
```

opinions

```
`ID` int(11)
`Text` text
`POSTags` text
`Creation` int(11)
`Author` varchar(64)
`Source` enum('hyves','twitter','facebook')
`Language` varchar(10)
`Subjectivity` float
`Polarity` tinyint(4)
`PolarityScore` float
`PolarSentence` text
`Emissions` text
`InProcess` varchar(6)
`StartProcess` int(11)
PRIMARY KEY (`ID`),
UNIQUE KEY `Creation` (`Creation`,`Author`,`Source`)
```

pos_tags

```
`ID` int(11)
`Text` text
`POSTags` text
`Creation` int(11)
`Author` varchar(64)
`Source` enum('hyves','twitter','facebook')
`Language` varchar(10)
`Subjectivity` tinyint(1)
`InProcess` varchar(6)
`StartProcess` int(11)
PRIMARY KEY (`ID`),
UNIQUE KEY `Creation` (`Creation`,`Author`,`Source`)
```

A.10 Data protocols

In this section we describe the data protocols that are used to communicate with the model interfaces residing in the *model* folder. There are five such interfaces, *front, mining, portfolio, work, store.* Any new function to be added to one of these interfaces should adhere to the protocols. Likewise, any file making use of one of the interfaces should adhere to these protocols as well.

All protocols should make a *POST* request to either of the interfaces. We list all *POST* fields that are required or are optional. We describe the intention of each field. Any return value is an associative array in JSON format.

A.10.1 Front

The front interface handles calls regarding the front-end. Any view-related functionality should be added to this interface. The protocol used by this interface is the following.

```
call: string
data: JSON
```

- **call** The call field is the function that should handle the call. This function should be defined in the front interface file.
- **data** The data field is a JSON-formatted field containing any data that the function triggered by **call** uses.

The following calls with their respective data fields are defined.

- **getOpinions** Obtains all opinions for a given term
  - Field: *terms – Required* The terms to fetch opinions for

A.10.2 Mining

The mining interface handles calls regarding any mining process. Any additional functionality required for mining purposes should be added to this interface. The protocol used by this interface is the following.

```
call: string
data: JSON
```

- **call** The call field is the function that should handle the call. This function should be defined in the front interface file.
- **data** The data field is a JSON-formatted field containing any data that the function triggered by **call** uses.

The following calls with their respective data fields are defined.

- **getLanguageTexts** Get texts that can be language identified
  - Array: *key: source, value: amount – Required* Sources to fetch texts of (one of *Twitter, Facebook, Hyves*) as keys. The amounts (integer) of messages to fetch as values
- **getPOSTexts** Get texts that can be POS-tagged
  - Array: *key: source, value: amount – Required* Sources to fetch texts of (one of *Twitter, Facebook, Hyves*) as keys. The amounts (integer) of messages to fetch as values
- **getOMTexts** Get texts on which subjectivity detection can be applied
  - Array: *value: array – Required* Every element is an array containing three fields
    - *source – Required* The source to fetch texts of (one of *Twitter, Facebook, Hyves*)
    - *amount – Required* The amount of texts to fetch
    - *language – Optional* The language of the texts
- **getPolTexts** Get texts on which polarity detection can be applied
  - Array: *value: array – Required* Every element is an array containing three fields
    - *source – Required* The source to fetch texts of (one of *Twitter, Facebook, Hyves*)
    - *amount – Required* The amount of texts to fetch
    - *language – Optional* The language of the texts

- **updateLanguage** Updates the language field of a text, as a result of language identification
  - Array: *value: array – Required* Every element is an array containing two fields
    * *ID – Required* The ID of the text to update the language of
    * *Language – Required* The new language

A.10.3 Portfolio

The portfolio interface handles calls regarding the term portfolio. Anything related to the portfolio should be handled through this interface. The protocol used by this interface is the following.

```
type: string
data: JSON
```

- **type** The type field is the type of action to take. A function corresponding to this type, appended with *Terms*, should exist in this interface.
- **data** The data field is a JSON-formatted field containing any data that the function triggered by **type** uses.

The following types with their respective data fields are defined.

- **get** Gets all terms in the portfolio, no data required
- **set** Sets the term portfolio
  - Array: *string – Required* An array containing the new terms. If the array is empty, no action is taken
- **facebook** Gets facebook terms to search for
  - Field: *amount – Optional* The amount of terms to fetch. If not specified, 100 terms are returned

A.10.4 Store

The store interface handles calls regarding the storing of any result. Any results modifying the database should be handled by this interface. The protocol used by this interface is the following.

```
type: string
source: string
data: JSON
```

- **type** The type field is the type of action to take.
- **source** The source from which the storage item originates, can be left empty. A function whose name is preceded by *store* and whose remainder is the concatenation of the *type* and *source* field should exist.
- **data** The data field is a JSON-formatted field containing any data that the function triggered uses.

The following types and sources with their respective data fields are defined.

- type **TermState**, source **Facebook** Stores the state of a Facebook term
  - Array: *string* – *Required* The state of all terms in the array is updated to 1
- type **AddTerms**, source **Facebook** Adds Facebook terms
  - Array: *string* – *Required* All terms in the array of at least 4 characters are added
- type **State**, source **Facebook** Updates the state of a Facebook message
  - Field: *ID* – *Required* The ID of the message
  - Field: *State* – *Required* The new state of the message
- type **UserScrape**, source **Facebook** Stores the result of a single user's scrape of Facebook
  - Field: *id* – *Required* The Facebook ID of the user
  - Field: *gender* – *Optional* The gender of the user
  - Field: *locale* – *Optional* The locale of the user
- type **PublicText**, source **Facebook** Stores a public text scraped from Facebook but only if the text contains more than 4 characters
  - Field: *text* – *Required* The content of the text
  - Field: *source* – *Required* The author (Facebook ID) of the text
  - Field: *creation* – *Required* The creation time of the text
- type **UserCrawl**, source **Facebook** Stores users crawled from Facebook
  - Array: *string* – *Required* The IDs of the Facebook users
- type **UserCrawl**, source **Twitter** Stores users crawled from Twitter
  - Array: *array* – *Required* The values contain the following, an ID should always be present and at least one of the remaining fields should be present
    - *id* – *Required* The user's Twitter ID
    - *location* – *Optional* The user's location
    - *description* – *Optional* The user's description
    - *language* – *Optional* The user's language
- type **PublicScrape**, source **Twitter** Stores the public tweets obtained after scraping Twitter
  - Array: *array* – *Required* The values contain the following
    - *user_id* – *Required* The author of the tweet
    - *text* – *Required* The tweet's contents
    - *creation* – *Required* The creation date and time of the tweet
- type **UserScrape**, source **Twitter** Stores user information obtained after scraping Twitter
  - Array: *array* – *Required* The values contain the following, an ID should always be present and at least one of the remaining fields should be present
    - *id* – *Required* The user's Twitter ID
    - *location* – *Optional* The user's location
    - *description* – *Optional* The user's description
    - *language* – *Optional* The user's language
- type **State**, source **Twitter** Updates the state of a Twitter message
  - Field: *id* – *Required* The ID of the Twitter message
  - Field: *state* – *Required* The new state of the message
- type **UserCrawl**, source **Hyves** Users crawled from Hyves

  - Array: *string* – *Required* The IDs of the Hyves users

- type **UserScrape**, source **Hyves** Stores user information obtained after scraping Hyves
  - Array: *array* – *Required* The values contain the following
    * *id* – *Required* The user's Hyves ID
    * *gender* – *Optional* The user's gender
    * *countryName* – *Optional* The name of the country the user lives in
    * *cityName* – *Optional* The name of the city the user lives in
    * *birthDay* – *Optional* The day part of the user's birth date
    * *birthMonth* – *Optional* The month part of the user's birth date
    * *birthYear* – *Optional* The year part of the user's birth date

- type **TextScrape**, source **Hyves** Stores scraps and WWWs obtained after scraping Hyves
  - Field: *id* – *Optional* Used for storing WWWs. The ID of the author of the WWWs
  - Array: *scraps* – *Required* The scraped scraps. Each element contains the following
    * *text* – *Required* The scrap's contents
    * *source* – *Required* The scrap's author
    * *created* – *Required* The creation date and time of the scrap
  - Array: *wwws* – *Required* The scraped WWWs. Each element contains the following
    * *text* – *Required* The WWW's contents
    * *created* – *Required* The creation date and time of the WWW

- type **SearchScrape**, source **Hyves** Stores WWWs obtained after searching for terms from the portfolio on Hyves
  - Array: *wwws* – *Required* The scraped WWWs. Each element contains the following
    * *author* – *Required* The WWW's author
    * *text* – *Required* The WWW's contents
    * *created* – *Required* The creation date and time of the WWW

- type **State**, source **Hyves** Updates the state of a Hyves message
  - Field: *id* – *Required* The ID of the message
  - Field: *state* – *Required* The new state of the message

- type **StoreLanguage**, source *empty* Stores the language of a text in the mining database
  - Array: *array* – *Required* All languages to store. Contains the following fields
    * *ID* – *Required* The ID of the message
    * *Origin* – *Required* The origin, must be one of *Twitter, Facebook, Hyves*
    * *Language* – *Required* The language of the message

- type **POS**, source *empty* Stores the parts of speech of a text in the mining database
  - Array: *Text* – *Required* The original text, contains the following fields
    * *ID* – *Required* The ID of the message
    * *Text* – *Required* The contents of the message
    * *Timestamp* – *Required* The timestamp the message was created
    * *Source* – *Required* The ID of the author of the message
    * *Origin* – *Required* The originating social medium of the message, must be one of *Twitter, Facebook, Hyves*
    * *Language* – *Required* The language of the message
  - Field: *POS* – *Required* The POS-tags of the message, must be an array where each element is a single line of the output of the TreeTagger

- type **Subjectivity**, source *empty* Stores the subjectivity of a text in the mining database
  - Array: *Array – Required* All texts to store the subjectivity of, contains the following fields
    - *ID – Required* The ID of the message
    - *Subjectivity – Required* 1 if the text is subjective, -1 if the text is objective, 2 if no subjectivity model exists for this language
    - *Text – Required* The content of the message
    - *POSTags – Required* The part of speech tags of the message
    - *Creation – Required* The creation time of the message
    - *Author – Required* The author of the message
    - *Source – Required* The source of the message, must be one of *Twitter, Facebook, Hyves*
    - *Language – Required* The language of the message
    - *SubjectivityScore – Required* The subjectivity score of the message
- type **Polarity**, source *empty* Stores the polarity of a text in the mining database
  - Array: *key: id, value: Array – Required* All texts to store the polarity of. The key is the ID of the text, the value contains the following fields
    - *Polarity – Required* 1 if the polarity is positive, -1 if the polarity is negative, 0 otherwise
    - *PolarityScore – Required* The real-valued score of the polarity algorithm
    - *PolarSentence – Required* Serialized version of the input of the RBEM algorithm for this message
    - *Emissions – Required* Serialized array containing the emissions per position for this message

A.10.5 Work

The work interface handles calls regarding work items. Any work items to be performed should be requested through this interface. The protocol used by this interface is the following.

```
type: string
source: string
data: JSON
```

- **type** The type field is the type of action to take.
- **source** The source from which the work item originates, can be left empty. A function whose name is preceded by *work* and whose remainder is the concatenation of the *type* and *source* field should exist.
- **data** The data field is a JSON-formatted field containing any data that the function triggered uses.

The following types and sources with their respective data fields are defined.

- type **GetUsers**, source **Facebook** Gets a list of users to crawl on Facebook
  - Field: *bagSize – Optional* The number of users to return
  - Field: *state – Optional* The state the users should have
- type **GetUsers**, source **Twitter** Gets a list of users to crawl on Twitter
  - Field: *bagSize – Optional* The number of users to return
  - Field: *state – Optional* The state the users should have
- type **GetUsers**, source **Hyves** Gets a list of users to crawl on Hyves
  - Field: *bagSize – Optional* The number of users to return
  - Field: *state – Optional* The state the users should have

B Figures of Mapping Sentiment Analysis Against Traditional Survey

This appendix lists the figures that lead to the conclusions drawn in Section 3.4. Appendix B.1 contains all figures used in Section 3.4.2 whereas Appendix B.2 contains all figures used in Section 3.4.3.

B.1 Natural Language Alignment

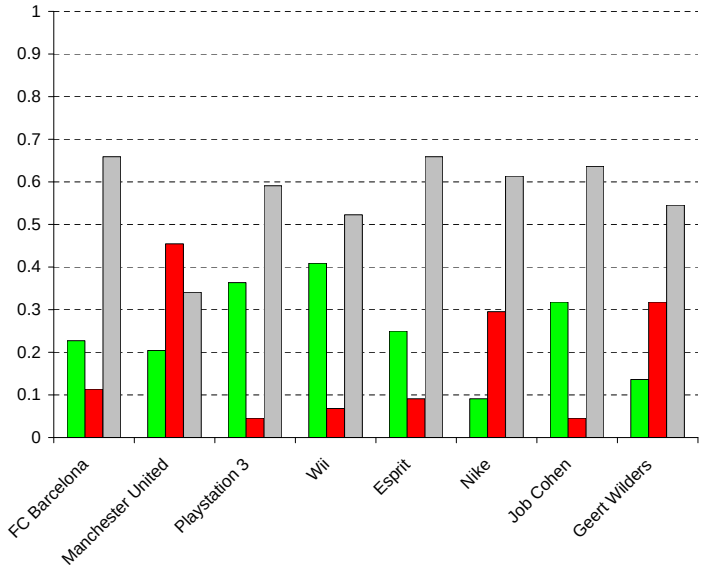


Figure 24: Sentiment distributions of our ground truth set. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. This figure should be compared against Figure 25.

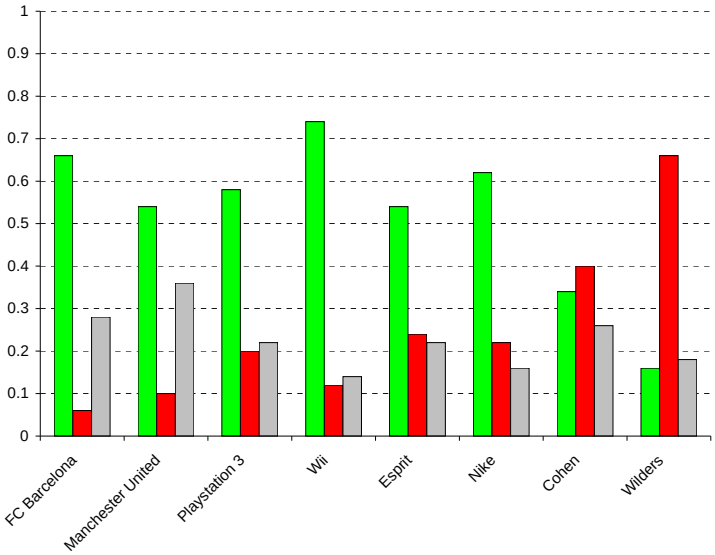


Figure 25: Sentiment distributions of our **Sample 2**. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. This figure should be compared against Figure 24.

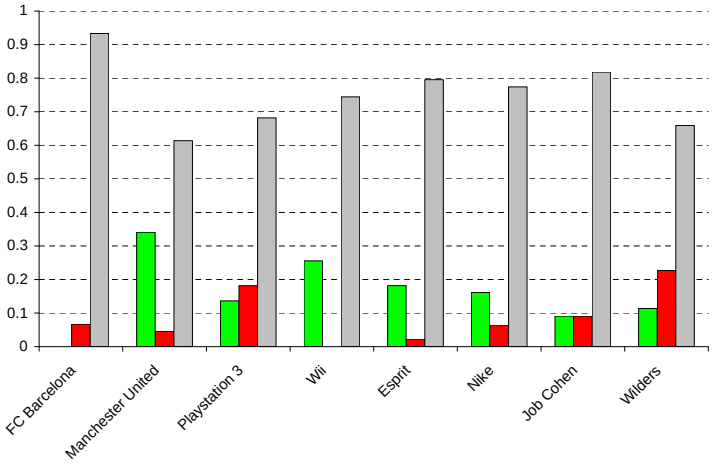


Figure 26: Sentiment distributions after running our sentiment analysis on the ground truth set. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. This figure should be compared against Figure 24.

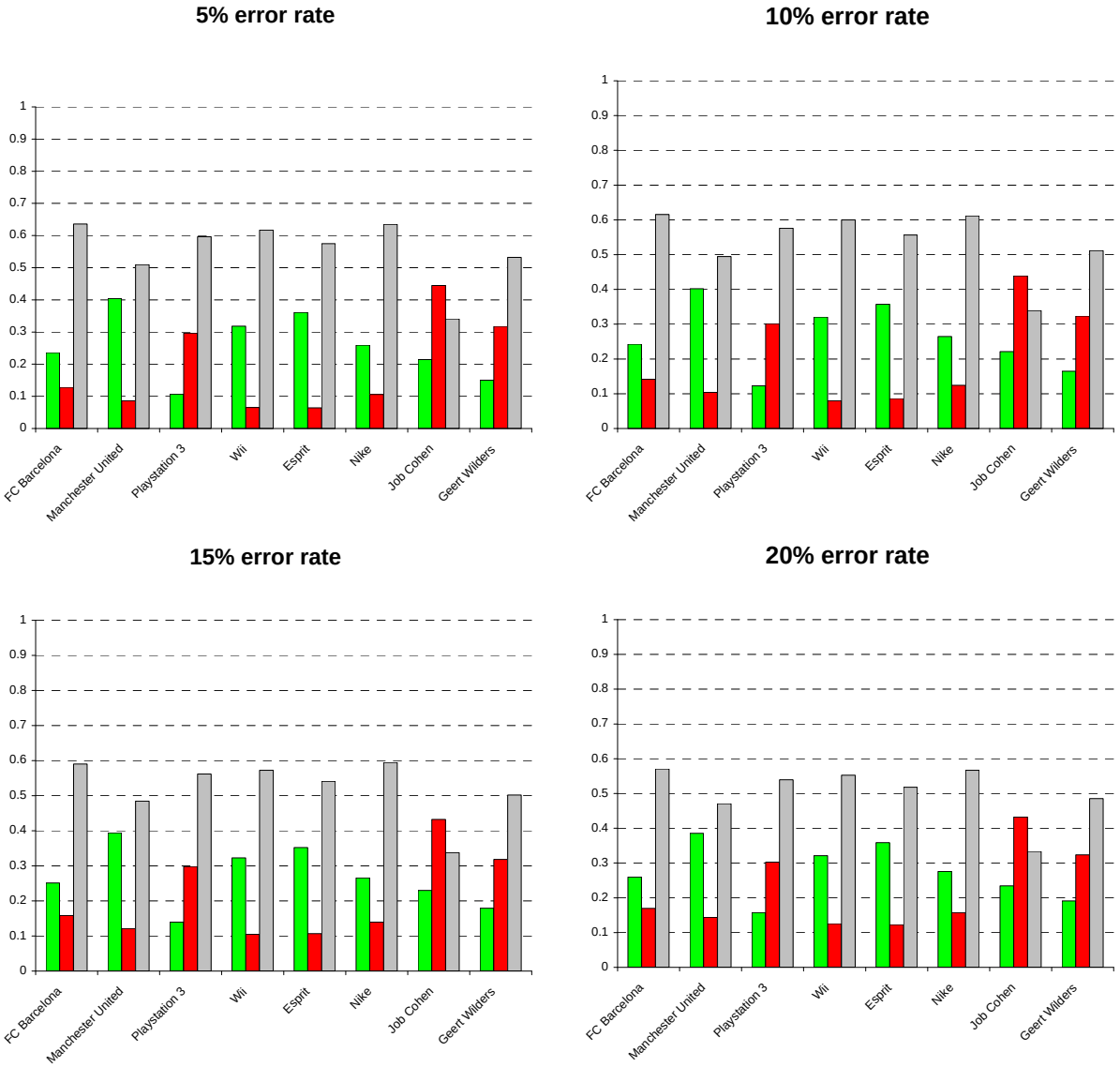


Figure 27: Sentiment distributions having an error rate of 5% through 20%. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. These figures should be compared against Figure 24.

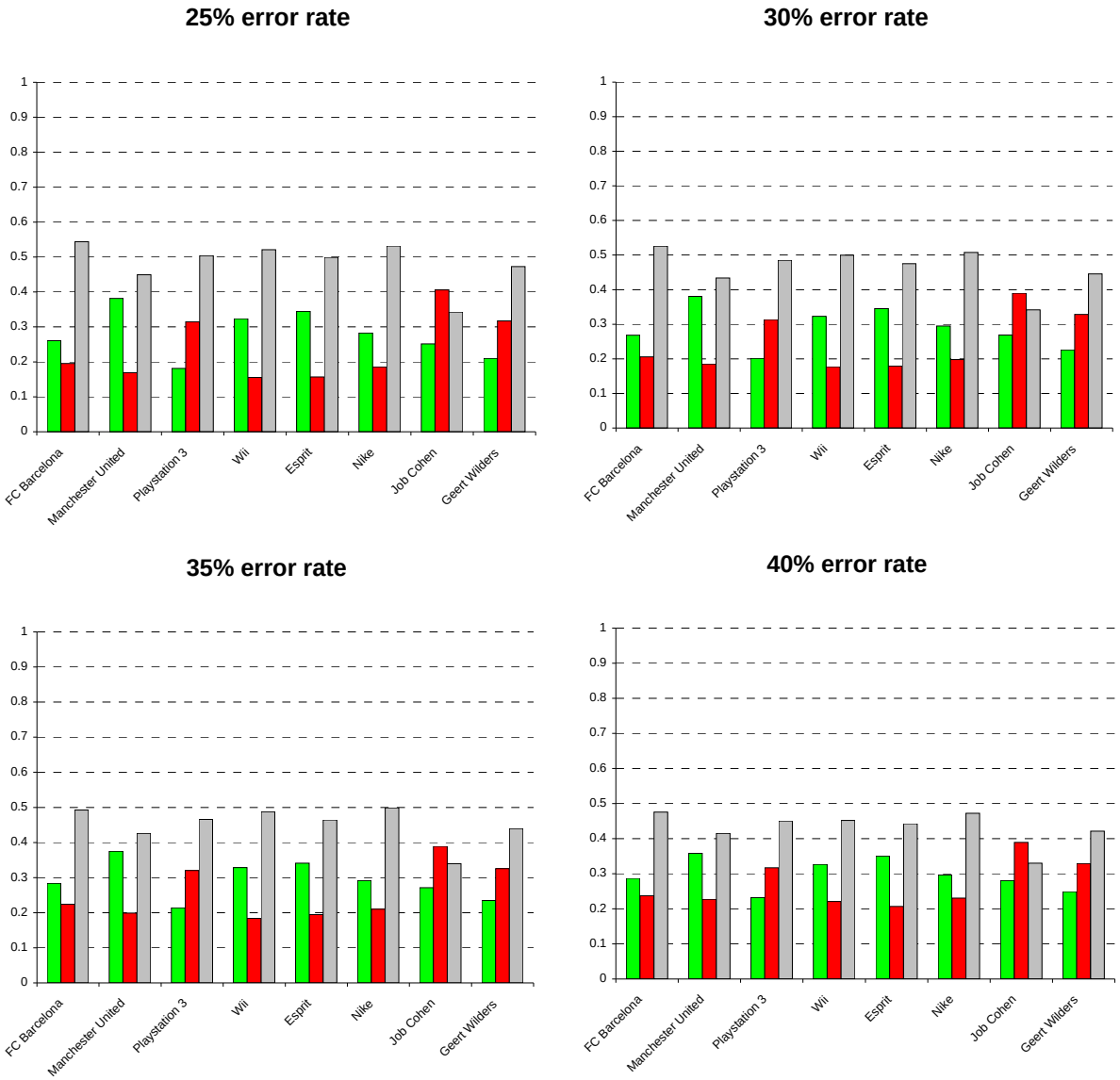


Figure 28: Sentiment distributions after running our sentiment analysis on the ground truth set. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. These figures should be compared against Figure 24.

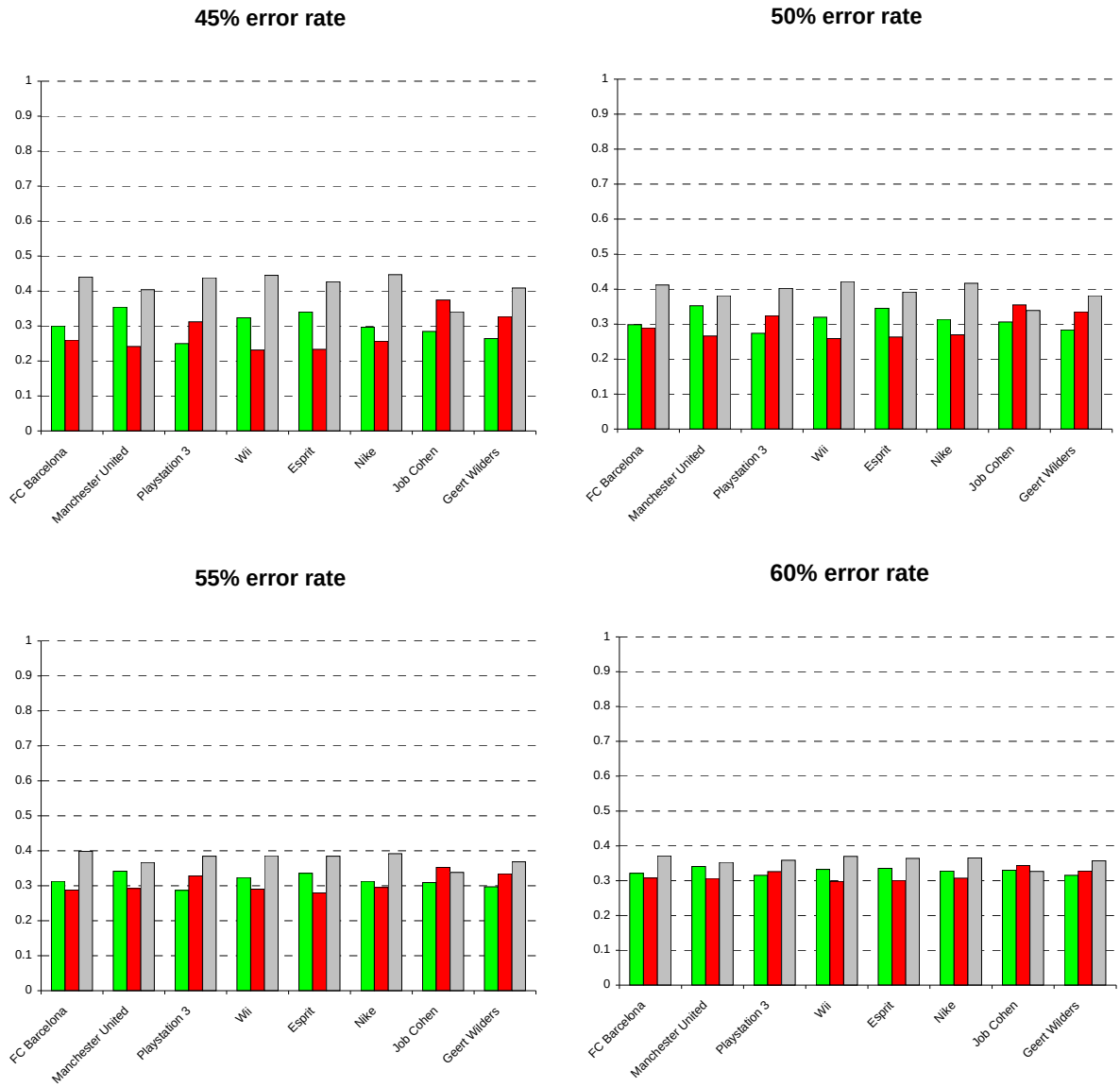


Figure 29: Sentiment distributions after running our sentiment analysis on the ground truth set. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. These figures should be compared against Figure 24.

B.2 Score-Based Alignment

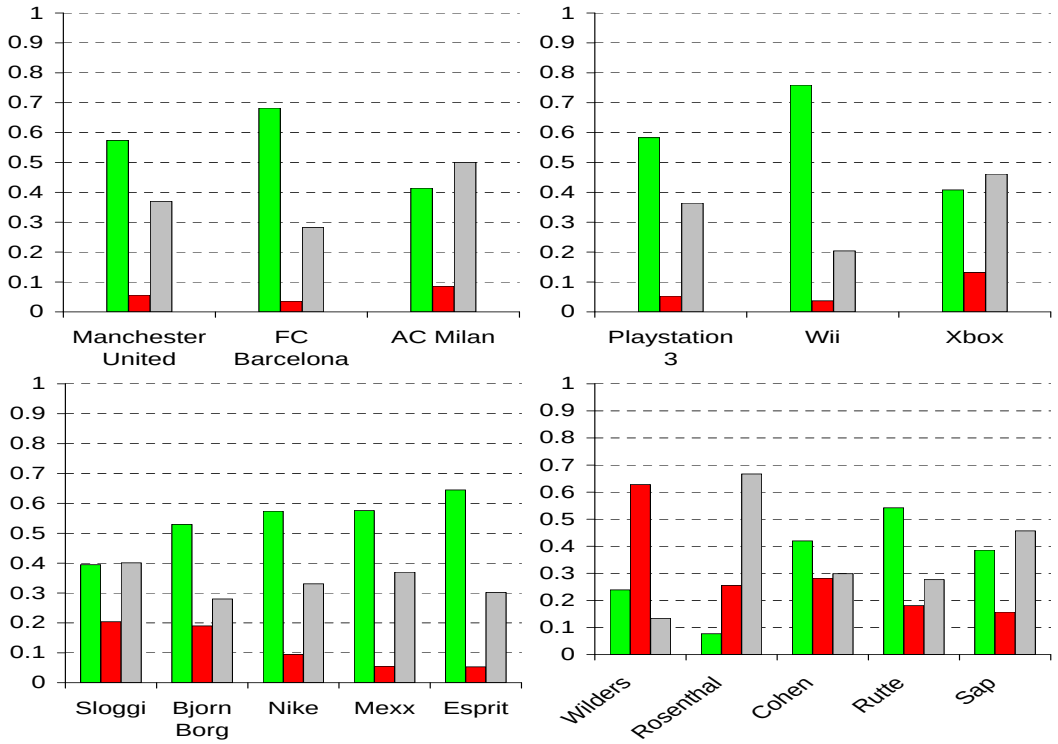


Figure 30: Aggregates of survey respondents using Hyves in age group [18, 29]. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. This figure should be compared against Figure 31.

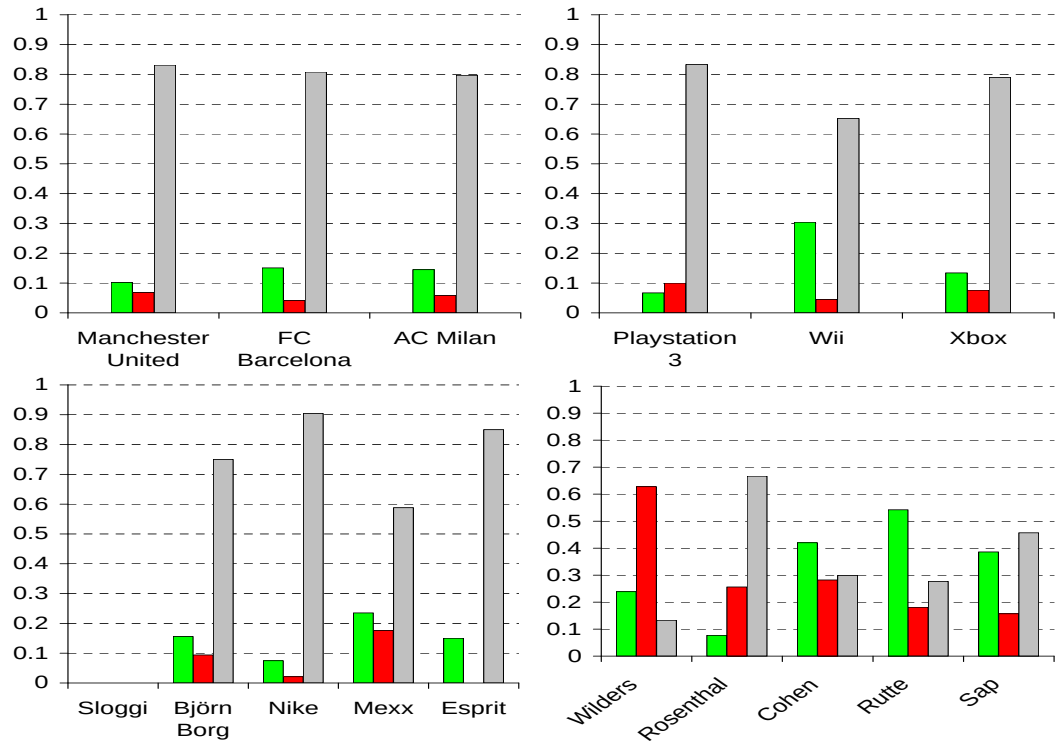


Figure 31: Aggregates of Hyves sentiment analysis in age group [18, 29]. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. This figure should be compared against Figure 30.

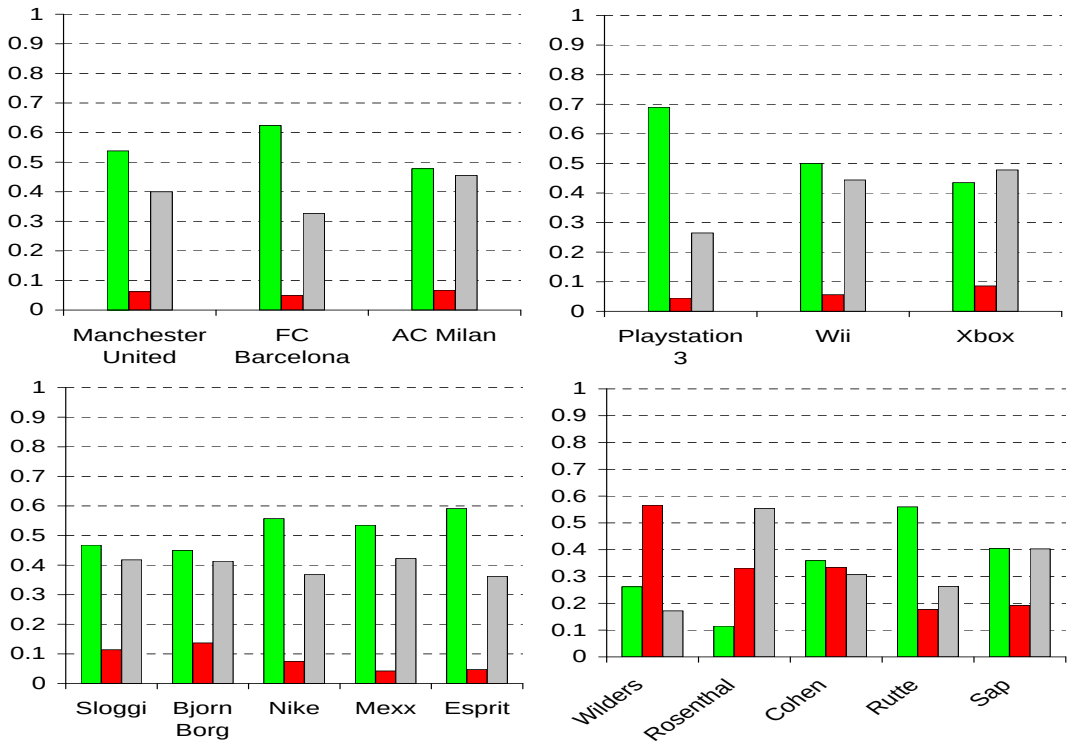


Figure 32: Global aggregates of survey respondents using Hyves. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. This figure should be compared against Figure 32.

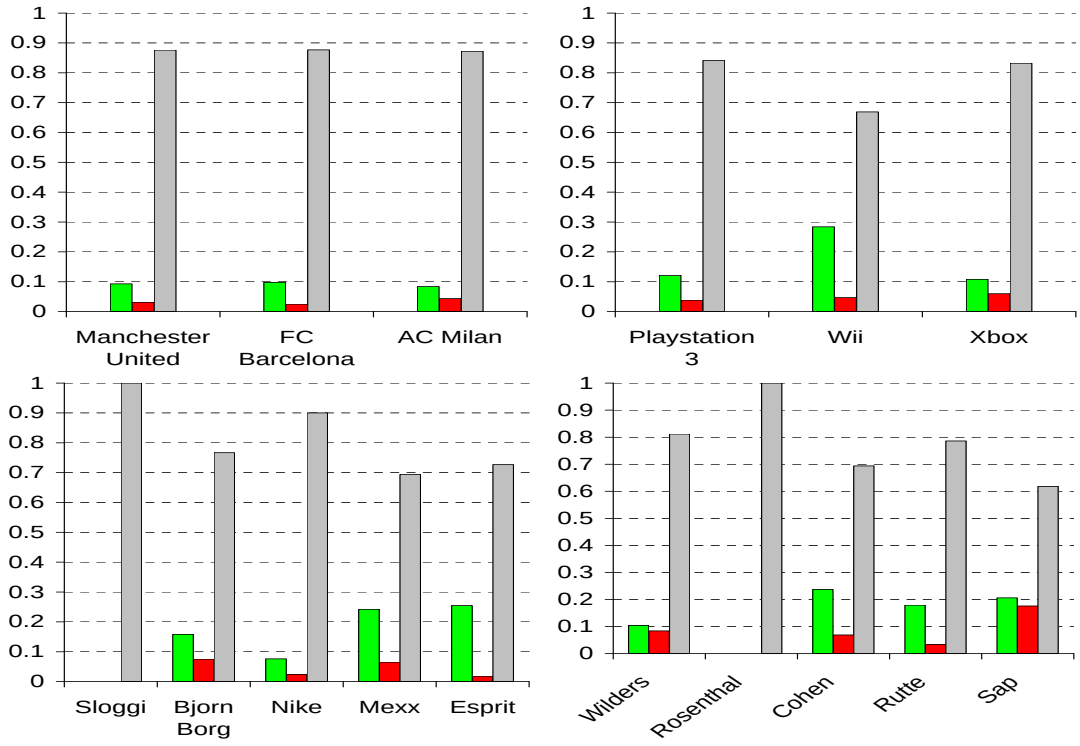


Figure 33: Global aggregates of Hyves sentiment analysis. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. This figure should be compared against Figure 32.

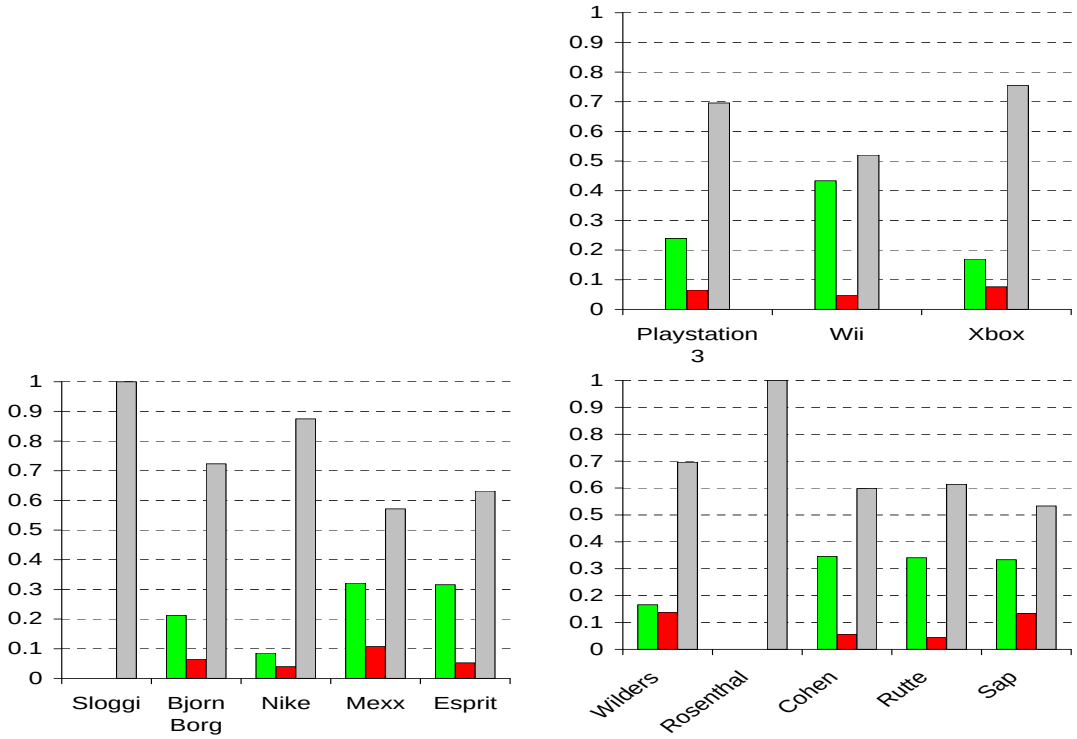


Figure 34: Biased aggregates of Hyves sentiment analysis. Green bars indicate the fraction of positive sentiment, red bars indicate negative sentiment and grey bars indicate neutral sentiment. This figure should be compared against Figure 32.

C Demonstrative Examples of Application of Sentiment Analysis

In our traditional survey experiments we stress the difficulties of making an alignment between sentiment analysis and traditional surveying. We state that the two are complementary. In this Appendix we will illustrate the complementarity by going through five demonstrative examples of applications of automated sentiment analysis in Appendices C.1 through C.5. We do not attempt to exhaustively present all strengths and weaknesses of sentiment analysis versus traditional surveying but show important application areas.

One of the main strengths of our sentiment analysis that applies to all examples we state now and do not repeat, is that the time required to obtain the results is almost instantly for our sentiment analysis but takes several days to weeks with a traditional survey. We do assume that the crawling and scraping of data for our sentiment analysis is continuously running but as this requires no human interaction, this is a realistic assumption. Another strong point of automated sentiment analysis versus traditional surveying is that with a traditional survey, each additional entity we would like to have information on requires us to actively ask for it whereas with sentiment analysis, we use what is already available. Moreover, getting sufficient answers for a traditional survey is often a problem as not enough people are willing to take the time to fill out a survey whereas for sentiment analysis it is easy to collect large amounts of data which are already present on social media.

With each example we will point out the capabilities our sentiment analysis gives that traditional surveying does not have and vice versa. For each example we briefly introduce the tasks we try to accomplish and sketch the setting in which this would be useful. We then use results that are produced by sentiment analysis to answer the tasks stated with each example. For consistency purposes, the illustrative examples used are on either of the entities presented in Table 7 used in our traditional survey experiments. Even though most of our entities are applicable in Dutch settings only, for illustration purposes we will use English examples. The resulting views we present with each of the examples do not show sentiment reflecting reality and should not be used to deduce statistical statements from.

Some of the views presented in these examples are actually implemented views present in our framework that we describe in Appendix A. These views can be automatically accessed and run through our framework. Other views are not implemented in our framework but manually constructed on the data present. These are thus legitimate views that can be created based on our data but currently are not, at least not in an automated fashion.

C.1 Sentiment Monitoring

Sentiment monitoring is a type of sentiment analysis performed over time, typically used to analyze the image of a brand or company. Though sentiment monitoring can have many application settings – in fact, all examples presented next require sentiment monitoring to some extend – we regard one particular setting. Consider a company selling season-bound products such as swim clothing, typically sold during the spring and summer. By the end of the summer the swim clothing is sold as bargains through which still sufficient money can be earned. The company wants to know how fond people are of the swim clothing as flexibly as possible to prevent any image damage and loss in sales and to be able to react quickly and possibly launch promotion campaigns targeting particular target groups that need additional stimulants. To make this example more concrete, we use the company *Mexx* as an example. This is the setting we regard, the concrete tasks of *Mexx* are the following.

Task 1 – Evaluate how well the public likes their season-bound swim clothing.

Task 2 – Monitor sentiment to quickly respond to peculiarities.

Both tasks directly relate to sentiment monitoring. For any type of sentiment monitoring, data needs to be collected and classified on a continuous base. This is not a difficult task for our sentiment analysis

but if we relate this to traditional surveying, monitoring continuously is tedious and often impossible as surveys typically only last for certain time period or only collect a predefined amount of responses.

The *Mexx* company mainly sells clothing but also perfumes, glasses and other types of fashion items. To make sure only the swimming clothing line of the company is being monitored, a query containing both the word *Mexx* as well as the word *swim* would be a good start to capture anything swim related that is applicable to our company of interest.

Mexx starts by monitoring sentiment entailing the term *Mexx* and *swim* before the summer period. An example view resulting from this is shown in Figure 35(a). From this view it can be seen that the general sentiment is more positive than negative. Throughout the summer, *Mexx* continuously monitors the sentiment which is shown in Figure 35(b). *Mexx* sees that during the start of the summer, more is being said about their swimming clothing, both positively and negatively. Positive sentiment is prevailing during the beginning of the summer but as July ends, negative sentiment is increasing and prevails.

Mexx gets alerted by the increase of negative around July 23rd and decides to take actions to revert the trend of increasing negative sentiment. They want to analysis which age group it is that they should focus on. To do so, an aggregation over time from the 14th of July up to the 23rd of July is made and segmented into age groups, resulting in Figure 35(c). Such a segmentation can only be done in traditional surveying if some sort of panel is maintained. From this segmentation, *Mexx* concludes that they should quickly start a promotion campaign targeting people in the age group 18-29 specifically aimed at decreasing negative sentiment.

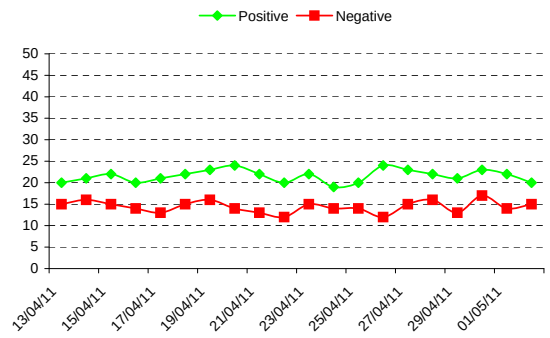
Once the campaign is launched it takes some time to influence the sentiment, which is clearly seen in the still increasing sentiment shown in Figure 35(b). The sentiment progresses as shown in Figure 35(d) in which the effect of the campaign is clear, showing an increase in positive sentiment. *Mexx* thus successfully reacted to a negative shift in sentiment and evaluated how the well the public likes their sentiment as the prevailing sentiment is positive except for the shift during the summer.

With this example we showed how automated sentiment analysis can be used to monitor sentiment, allowing for an instant insight into segments if a shift in sentiment is noticed. We list the differences between sentiment analysis and traditional surveying related to this example.

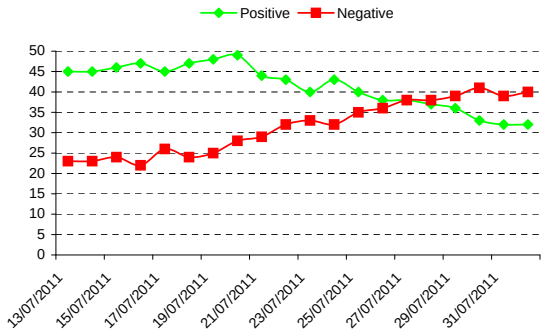
- The continuous monitoring of sentiment is something that is typically not possible in traditional surveying as surveys only last a given time. Using traditional surveying, *Mexx* could have periodically – say every week – checked the sentiment by surveying respondents. Moreover, every new period, a new survey has to be distributed and analyzed whereas with automated sentiment analysis, monitoring is a continuous process not requiring any human intervention.
- If segmentation is to be performed, *Mexx* would need to maintain a panel. This is typically a costly aspect not required with automated sentiment analysis as all information upon which we can segment comes without additional costs.
- With traditional surveying, *Mexx* could directly ask respondents exactly what it is about their swim clothing that people do not like. With our sentiment analysis, this is not possible in an automated way. Though we could extend our sentiment analysis to identify components in messages to which the sentiment relates, this is currently not done. With traditional surveys however, this process is also not automated but the respondent could be given a predefined list of answer options (and an additional escape which would have to be manually inspected) which then allows for easy aggregation.

C.2 Benchmarking

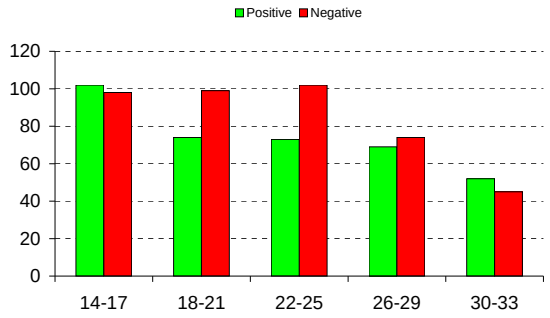
Benchmarking is the process of comparing one particular process or results thereof against comparable results of another process or best practices. This allows a company or brand to compare its own sentiment



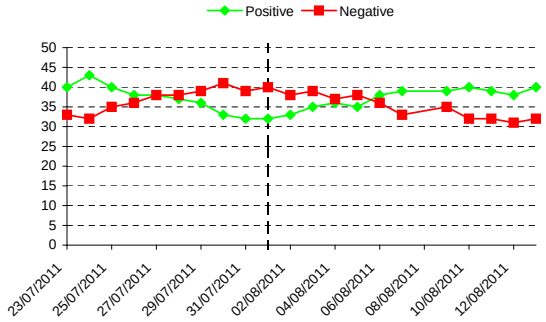
(a) A sentiment monitoring view for *Mexx* before the summer period.



(b) A sentiment monitoring view for *Mexx* during the summer period.



(c) An aggregation from July 14th up to July 23rd, segmented into age.



(d) A view showing the sentiment at the end of the summer. The vertical line is where the view of Figure 35(b) stopped.

Figure 35: Different sentiment monitoring views.

against that of others, for example competitors. When segmentation is possible, one can moreover gain insights into which segments favor your company and which segments favor others.

We consider a company trying to keep up in an innovative and relatively new market in which obtaining market segment is critical for a prosperous future. The concrete market we regard is that of gaming consoles. We take the perspective of *Sony* producing the *Playstation 3* gaming console. Their direct competitors in the gaming market are *Microsoft*'s *Xbox* and *Nintendo*'s *Wii*. *Sony* wants to know how well their *Playstation 3* is performing in this market with respect to these competitors. More precisely, the task *Sony* wants to achieve is the following.

Task 1 – Find out how well *Playstation 3* performs in the gaming console market with respect to its direct competitors.

For this benchmark, not only the console name but also the company name behind the console is of importance. This is because sentiment on a product of a company typically relates to the sentiment on the company itself and vice versa. We thus need to take messages into account that contain either of the company names *Microsoft, Sony, Nintendo* as well as the actual console names *Xbox, Playstation 3, Wii*.

The way to do this type of benchmarking is relatively easy. In case of our automated sentiment analysis, one just has to compare sentiments on the three (or six if we count in company names) different search terms. With traditional surveying the same can be done but requires more work. A survey posing questions on all three game consoles has to be created in which it is important that type of questions asked to obtain sentiment on a single game console is directly comparable to the type of question asked to obtain sentiment on the other consoles.

A possible view that allows for easy benchmarking is shown in Figure 36. This view shows the difference in positive and negative sentiment, computed simply as *positive - negative*. All values are above 0, indicating

that the sentiment on all three entities is prevailingly positive. It clearly shows that the *Wii* enjoys the best sentiment and that *Playstation 3* performs slightly better than *Xbox* with respect to sentiment.

In addition to such a benchmark, we can look at the view shown in Figure 37. This figure also shows a clear benchmark but now a more fine-grained one where we can see the difference in sentiment for different age groups. This segmenting shows that especially with people aged over 30 are not as fond of a *Playstation 3* than they are of a *Wii*. For *Sony* these benchmarks could give rise to marketing actions in which their *Playstation 3* directly competes with *Xbox* in the lower age groups under 30 and where they look at the *Wii* to borrow their ideas to target new segments in age groups over 30.

With this example we showed how easily one can benchmark against other entities or brands. On top of the already mentioned strengths of using sentiment analysis in general, the only differen between using sentiment analysis rather than traditional surveying is an aspect not pointed out in this example but rather in the example presented in Appendix C.1 which is the following.

- Not only can we easily create a benchmark as we have seen in this example, we can also monitor benchmarks to create periodic reports showing changes in sentiment after for example a new campaign of *Sony* to target other age groups. This pro is not so much a benchmarking aspect but rather a monitoring one.

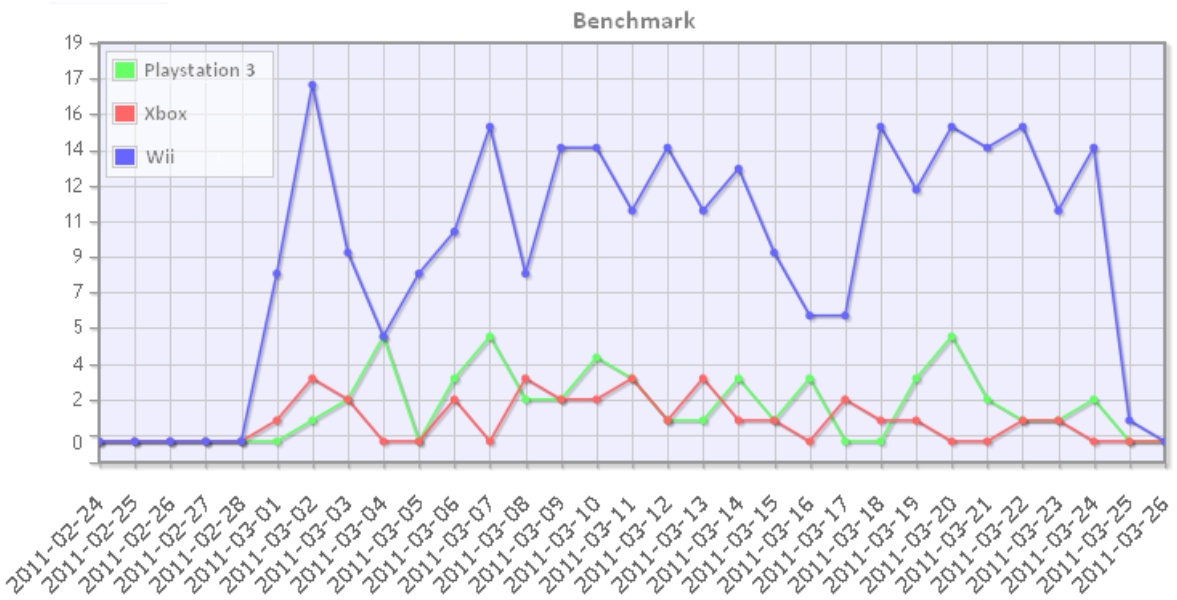


Figure 36: A benchmark view showing the difference in positive and negative sentiment. This easily shows that *Wii* is the best talked about whereas *Xbox* is the least.

C.3 Public Polling

Public polling is something typically done using traditional surveys. Through public polling, people want to know what the attitude of a group of people – typically a nation – is on a specific topic. A typical example of public polling is with any sort of elections where public polling is used to get preliminary insights into how parties perform during the elections.

As mentioned, public polling is typically performed using traditional surveys. Recently however, [O'Connor et al., 2010] performed a study on performing public polling through prior polarity sentiment analysis on Twitter. One of their examples was on the presidential elections of the United State of America of 2008 where public polling was performed using sentiment analysis. The focus was on the two main runners for the presidentship, Obama and McCain.

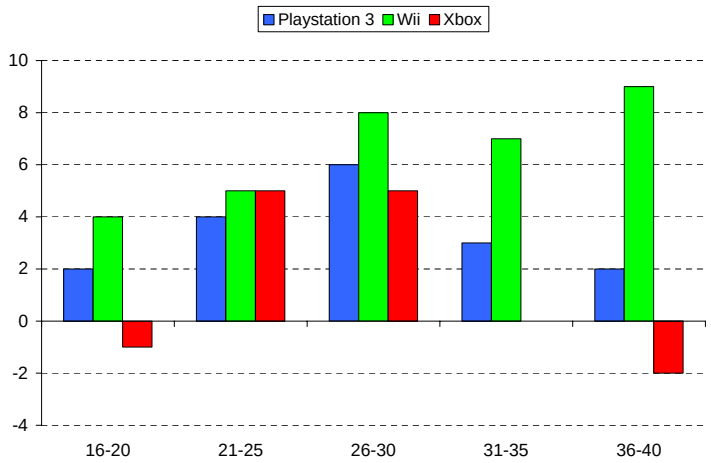


Figure 37: A benchmark view showing the difference in sentiment for different age groups.

Though [O'Connor et al., 2010] performed polling on Twitter and in an US-oriented fashion, our case will regard a Dutch setting. We consider a parliament election in the Netherlands in which we publicly poll the politicians used in our earlier experiments: *Geert Wilders, Mark Rutte, Job Cohen, Jolande Sap, Uri Rosenthal*. Being members of differently oriented parties, we hope to gain insights into which of the parties these politicians belong to will do better than others as the elections approach. We thus again need to continuously monitor as we want to keep an up-to-date view on matters as the elections close in. The task of this example is not a general polling goal but a more specific one targeting elections. This goal is the following.

Task 1 – Continuously perform public polling to get an up-to-date view on the parties' stances for the elections.

One of the major problems that arises with any public polling is that in order to obtain any trustworthy statistics, the poll needs to be held amongst a representative sample of people. Knowing what such a representative sample is, is not the problem as we can use numbers from the CBS that define the distribution of age of the Dutch population. The problem however is to find the required amount of respondents for each such age group. With traditional surveying this is not so cumbersome if we maintain a panel but for sentiment analysis we have seen that mainly low age groups are well represented on social media whereas high age groups are almost not represented at all.

Assumed that we have the ability to construct a representative sample of Dutch society. As Hyves is the only social network that allows us to extract age, we are bound to use that social medium. We can now use views such as those used in our traditional survey experiments to compare the politicians with each other. An example of such a view is given in Figure 38. For this view, we collect all sentiment on all politicians and create age groups $g_1, .., g_n$ as defined by the CBS in a per-politician fashion. We now take the age group that is worst represented, that is, the age group g_i that has the least amount of opinions o_i on a politician. We then set then set the fraction of which this age group should appear to be representative as defined by the CBS to translate to o_i. Using o_i as a reference point, we can now define how many responses we need for the other age groups in order to construct a representative sample. We only maintain those sentiment messages of which the author is in our representative sample, the other messages are discarded. The resulting view shown in Figure 38 is thus representative for Dutch society.

From Figure 38 we can see that currently *Wilders* is performing worst whereas *Rutte* is performing best. This view is a single poll. Polls usually encompass a certain time span. In addition to using only messages created by members of our representative sample, we thus also use only those messages that have been created in the last x days, where x is our timespan. This way, we always an up-to-date poll that shows the current sentiment rather than sentiment that is influenced by phenomena that occurred a long time ago and thus are no longer of influence or only of little influence.

With this example we showed that sentiment analysis can be used for public polling. More specifically, we showed how sentiment analysis can be applied for election polls. We identified a problem that whenever we want our poll's results to be representative, we need to have a representative sample of a nation or society. We list the strengths and weaknesses of using sentiment analysis to perform polling rather than using traditional surveys.

- For representative polling, we require a sample of Dutch society that is representative. As social media are typically heavily used by younger people rather than older people, we may have a problem to get a large enough representative sample that also encompasses the higher age groups. With traditional surveying, this problem is not as strong. Moreover, from our experience with all responses to the survey used in our traditional survey experiments in Section 3.4, elderly people seem more willing to answer than younger people but obtaining a representative sample is not a big problem.
- Even though mentioned as a general strength of our sentiment analysis, we once again stress the importance of continuity here. Where traditional surveying needs to be performed periodically to obtain an up-to-date view, our sentiment analysis can be ran continuously and thus show shifts in polls more quickly. We stress the possibility of continuity with sentiment analysis as this is something that was not possible before.

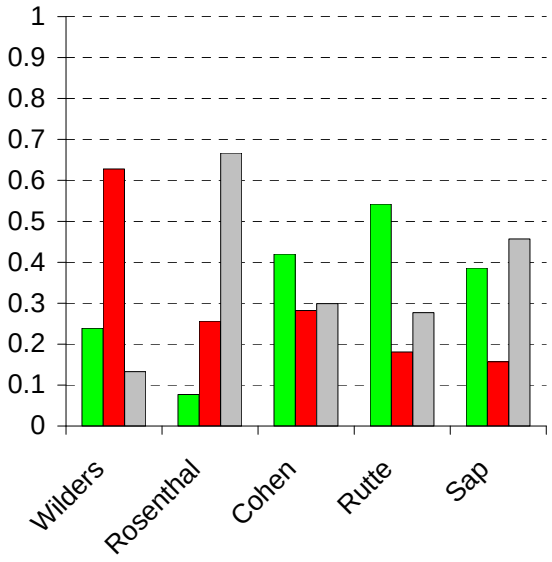


Figure 38: A view allowing to poll politicians for the elections.

C.4 Sentiment Forecasting

We can use sentiment analysis of today to forecast the future trends of tomorrow. Forecasting can be applied in many settings. To list two of them, we could use forecasting to predict how the trend of increasing negative sentiment would evolve in the monitoring example discussed in Appendix C.1 or we could use forecasting to predict the actual election outcome in our public polling example discussed in Appendix C.3. For this example on forecasting, we will expand upon the latter example where we polled elections to predict the actual election outcome. The task we want to achieve is the following.

Task 1 – Forecast future sentiment based on sentiment analysis results obtained thusfar.

A typical way to forecast feature trends based on the past is to use regression analysis. Regression analysis includes modeling and analyzing variables as other variables vary. In our case, we would model sentiment as a variable and analyze what happens as time, another variable, changes. As we use time as

our varying variable, we can use regression to make statements about future development in sentiment. In this example, we do not attempt to extensively explain and discuss regression and all of its application possibilities but rather see it as a tool to create forecasting models.

If we plot the positive sentiment of a single politician, for our example we will use *Wilders*, we can learn and apply a regression model on this positive sentiment to predict future sentiment. A view showing this is presented in Figure 39. The view is created on March 19th to include all sentiment of previous days of the month March. We thus have sentiment up to March 18th. The green dots show the positive sentiment on *Wilders* up to March 18th. The red line represents a linear regression model learned on the green dots. As we can see, a minor declining trend is visible. Predictions for upcoming days are hence that the positive sentiment on *Wilders* will decline slightly. If the elections are to be held within for example three days, we can use our regression model on top of public polling to not only state that *Wilders* is performing worst in the polls (as was concluded in the example in Appendix C.3) but will likely perform even worse at the actual elections.

Note that the example presented with this example is purely illustrative. In practice, such a simple and straightforward linear regression model may not be useful when more complex trends are present but for the sake of illustration this example suffices. The conclusion drawn based on the regression is also not as hard as we present it as errors and reliability play an important role with regression analysis, these aspects do need to be taken into account in an application setting.

In this example we showed how sentiment analysis can be used to forecast future sentiment using additional techniques such as regression analysis. We showed how polling a politician advancing to elections can be used to forecast actual election outcomes. On top of the general motivation of using sentiment analysis over traditional surveying, this example does not point out any additional strengths and weaknesses since forecasting is performed using different techniques such as regression which are not influenced from the way the data is collected. We can thus use either sentiment analysis or traditional surveying to obtain the data.

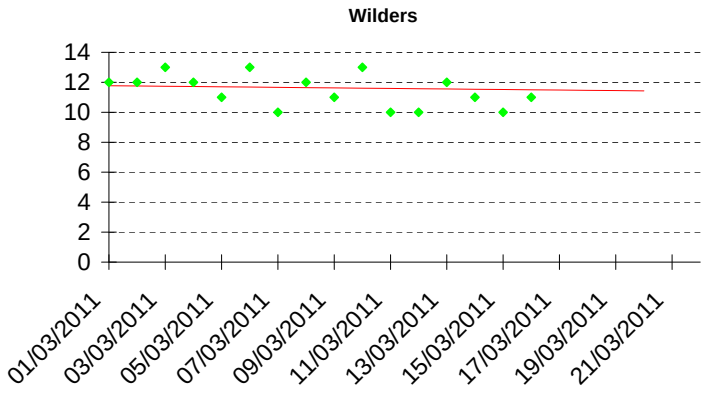


Figure 39: A forecasting view for *Wilders*. The green dots are the positive sentiment occurrences whereas the red line is a linear regression prediction.

C.5 Customer (Web) Care

Customer care is a type of customer relationship management (CRM). A traditional and typical part of customer care used to be support desks where customers of a company could call or write to to receive support from professionals or file complaints to to express their dismay. Nowadays, support desks are disappearing as customer care shifts to the internet. Nowadays people use e-mail rather than written mail or even a phone call to contact aa company. More recently people are even contacting Twitter accounts of customer web care services of companies to express their liking, disliking or to receive support of any kind. Companies appoint employees to perform this kind of web care dedicated. Finding and going through all messages that are being placed on social media that relate to the specific company is an impossible task and hence automated tools need to be used to support these web care services. The tasks we thus want to achieve are the following.

Task 1 – Support in finding out and analyzing where and for whom customer care is required.

Task 2 – Support in actually providing this customer care.

In order to find out where customer care is required, we first need to define when customer care is required. In this example, we will assume that customer producing negatively oriented messages potentially require customer care. Customers producing positively oriented messages are already satisfied and even though reaction to these type of messages may be beneficial, we do not pay attention to those. Neutral messages are also not regarded as, similar to what we assumed for our sentiment analysis, these are assumed to coincide with objective messages. Though objective messages may point out deficiencies of a company, they usually do not involve customers attitude towards a company and hence do not require customer care.

For this example we again consider *Sony*. More specifically, we consider a dedicated web care team responsible for customer care of *Playstation* customers, including the game console *Playstation 3* but also any other *Playstation* product. To achieve our first task, the web care team needs to be able to quickly find where customer care is required. As our strategy determined that negatively oriented messages require our attention, we need to able to find negatively oriented messages quickly.

The view shown in Figure 40 filter individual messages that contain negatively oriented sentiment. It only shows negatively oriented messages and allows to sort the messages either by showing the newest first, allowing to quickly respond and make sure the negative sentiment does not reach other customers or to by showing the oldest first, allowing to prevent customers to wait a long time or indefinitely before any attention is paid to their complaint or question. We could additionally order the messages based on the level of negativity by using the score the RBEM algorithm assigns to each messages but for this example we do not consider this.

Now that the web care team has found messages that require customer care and that the web care team is supported in analyzing the messages by the bright red coloring, actual customer care can be provided. The view shown in Figure 40 shows a 'Reply' link with each message. This link allows the web care team to directly respond on the message. In case of Twitter this is done by using a mention through the @ symbol and the name of the author of the message. In case of Hyves and Facebook a direct reply is made. The only constraint is that the web care team needs to be logged into their web care accounts for each of the social media as these accounts will be used as authors of the reply.

When the customer care team clicks on the 'Reply' link of a message, a dialogue appears where a reply can be entered. This is shown in Figure 41. The customer care team can then either decide to actually reply or to cancel. When a reply is made, the status of the message is updated to indicate a reply has been sent, preventing duplicate work. This way, the web care team is supported in customer care by allowing them to quickly respond to messages.

This example illustrated how sentiment analysis can be used to support customer care teams in performing web care. We showed how sentiment analysis can identify what messages require attention and quickly gain insights on what part of the message to focus on. A direct reply option allows the customer care team to quickly respond and more efficiently perform web care activities.

Listing the strengths and weaknesses of using sentiment analysis with respect to using traditional surveying makes no sense for this example due to the simple fact that customer care the way we presented it in this example is not possible through traditional surveys at all. This example thus illustrates a possible application of sentiment analysis that was previously not possible.

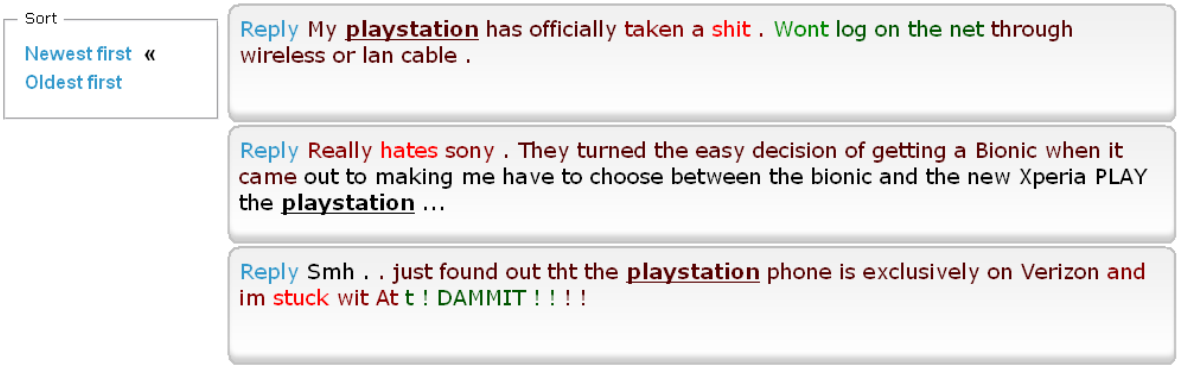


Figure 40: A view showing all the negatively oriented messages. The brighter red a word is, the more negativity it expresses, as per the emissive model of our RBEM algorithm described in Section 2.4.2.

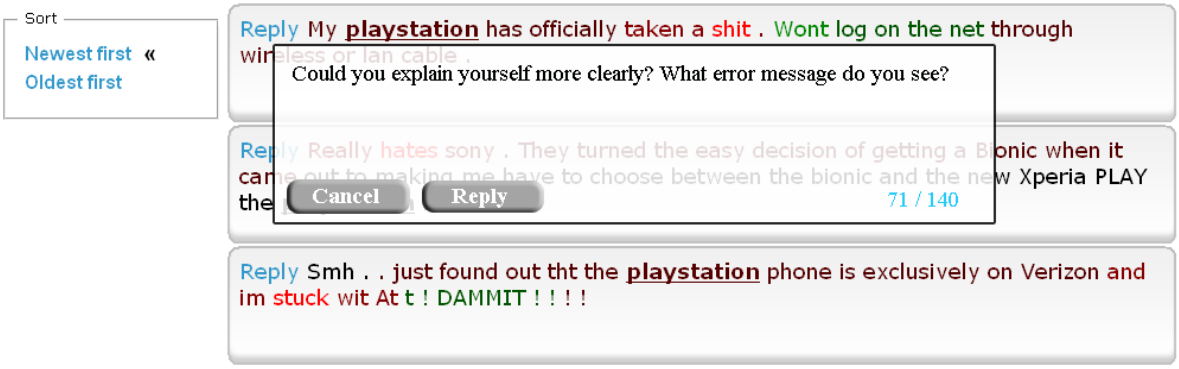


Figure 41: Replying to negative messages. A dialogue appears with the message to reply on.

D Survey

The original survey was held among Dutch citizens. Its questions are hence posed in Dutch. We present a translated version below, stating each question with its respective answer options. For each answer option, we showed a logo representing the entity or a picture of the person when applicable.

1. On which of the following social media are you active? (multiple answers possible)
   (a) Hyves (yes/no)
   (b) Facebook (yes/no)
   (c) Twitter (yes/no)
2. How interested are you in soccer? (only one answer possible)
   (a) Very interested
   (b) Quite interested
   (c) A little interested
   (d) Not really interested
   (e) No interest whatsoever
3. Please rate the following soccer clubs from 1 (very good) to 5 (very bad)
   (a) Manchester United (1..5, no opinion)
   (b) FC Barcelona (1..5, no opinion)
   (c) AC Milan (1..5, no opinion)
4. How interested are you in (clothing) brands? (only one answer possible)
   (a) Very interested
   (b) Quite interested
   (c) A little interested
   (d) Not really interested
   (e) No interest whatsoever
5. Please rate the following (clothing) brands from 1 (very good) to 5 (very bad)
   (a) Sloggi (1..5, no opinion)
   (b) Björn Borg (1..5, no opinion)
   (c) Nike (1..5, no opinion)
   (d) Mexx (1..5, no opinion)
   (e) Esprit (1..5, no opinion)
6. How interested are you in game consoles? (only one answer possible)
   (a) Very interested
   (b) Quite interested
   (c) A little interested
   (d) Not really interested
   (e) No interest whatsoever
7. Please rate the following game consoles from 1 (very good) to 5 (very bad)
   (a) Sony Playstation 3 (1..5, no opinion)
   (b) Nintendo Wii (1..5, no opinion)
   (c) Microsoft Xbox 360 (1..5, no opinion)

8. How interested are you in politics? (only one answer possible)
   (a) Very interested
   (b) Quite interested
   (c) A little interested
   (d) Not really interested
   (e) No interest whatsoever
9. Please rate the following politicians from 1 (very good) to 5 (very bad)
   (a) Geert Wilders (1..5, no opinion)
   (b) Uri Rosenthal (1..5, no opinion)
   (c) Job Cohen (1..5, no opinion)
   (d) Mark Rutte (1..5, no opinion)
   (e) Jolande Sap (1..5, no opinion)